Diagnosis and Reconfiguration Planning With Resource Constraints

James Kurien

Embedded, Collaborative Computing Area

Palo Alto Research Center

Worked performed at:

NASA Ames Research Center under 632 funding

Palo Alto Research Center

in collaboration with:

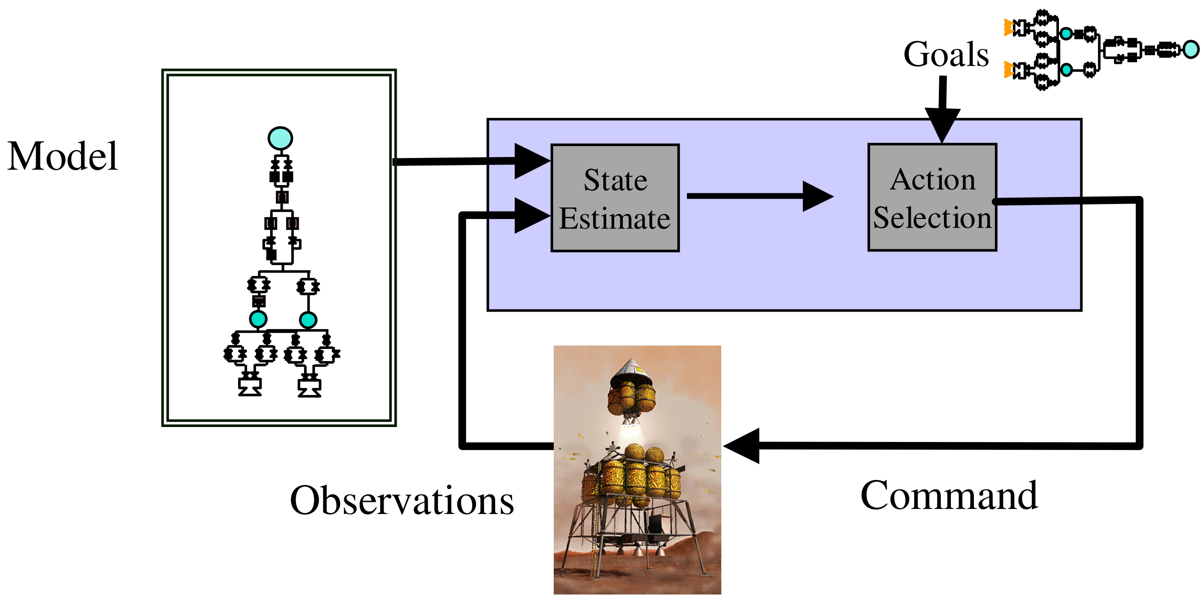
Pandu Nayak, David E. Smith

Lee Brownston

parc
Palo Alto Research Center

Problem Statement

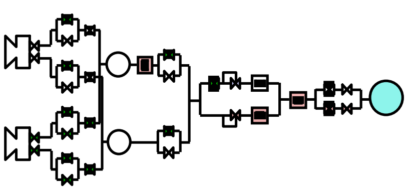
- Given
  - A model of a physical system such as a printer or spacecraft
  - The internal actions taken and observations received thus far
  - A description of the desired state of the system



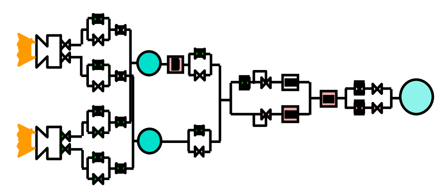
- Task
  - Determine the most likely internal states of the system
  - Find commands to move any likely state to a desirable state
  - If that’s not possible, do the best you can

Challenges

Initial State



Goal States

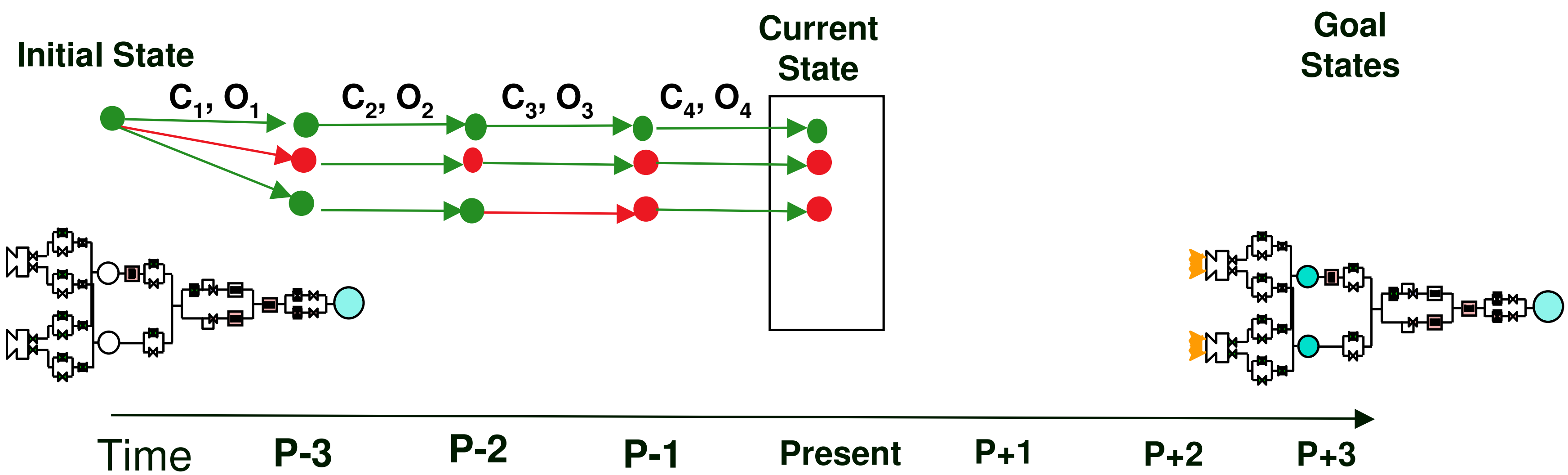


Time

parc Palo Alto Research Center

Challenges

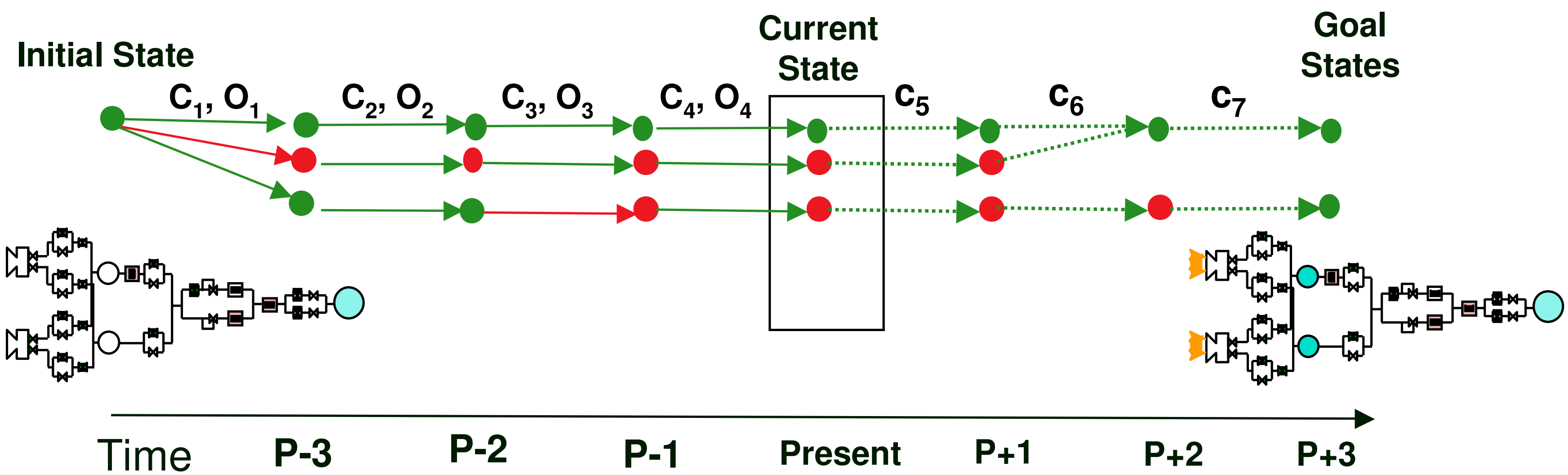
- Diagnosis or State Estimation
  - **Definition:** After each command, determine the **set** of likely states
  - **Problem:** The state is not completely observable. The number of states is huge.
  - **Problem:** Failures may not manifest themselves at the time they occur.



parc Palo Alto Research Center

Challenges

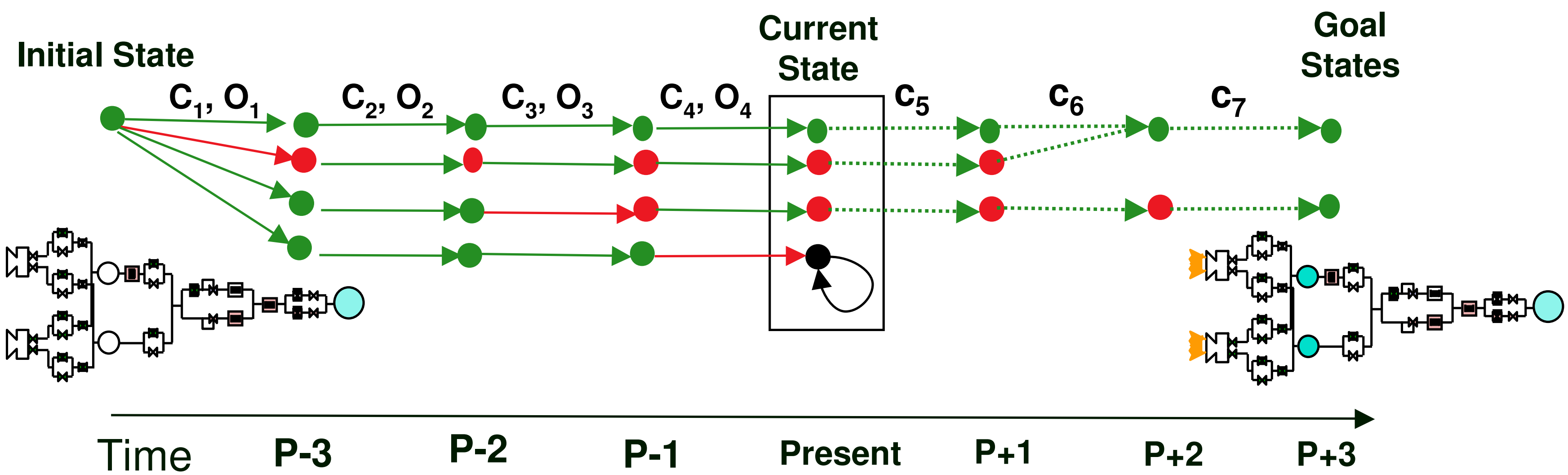
- Conformant Planning
  - **Definition:** Given a set of states, find one plan that achieves the goal in every state
  - **Problem:** Actions chosen for one state can have unintended effects in another.



parc Palo Alto Research Center

Challenges

- Planning With Failures
  - **Definition:** Plan to achieve as much as possible given failures and time limits
  - **Problem:** Every goal may not be achievable in every likely state.
  - **Problem:** Some combinations are much more difficult to plan for than others.

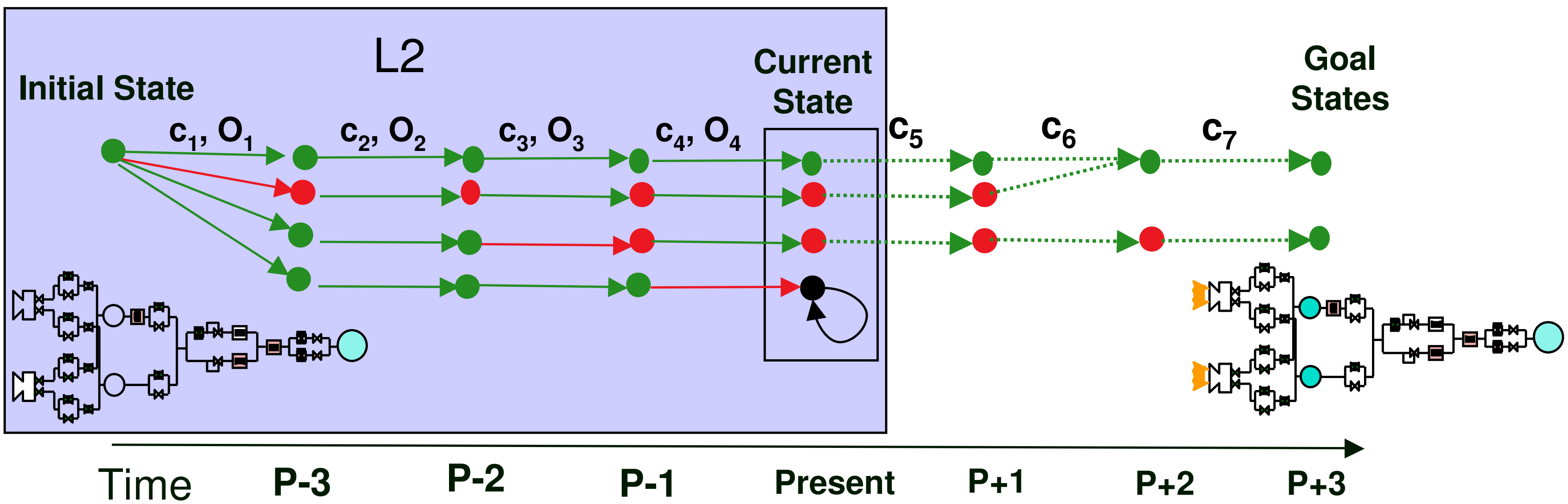


parc Palo Alto Research Center

Status

- Diagnosis or State Estimation: L2 (Kurien & Nayak, AAAI 200)
  - Tracks multiple system trajectories
  - Backtracks to find failures that were not immediately observable
  - Extends ideas of Livingstone (Williams and Nayak, AAAI 1996) as flown on DS1
  - Used by S/C engineers to develop X-34, X-37 models & diagnostic scenarios
  - In use at NASA, licensed to a spacecraft software company
- Conformant Planning: fragPlan (Kurien, Nayak & Smith, AIPS 2002)
  - Novel, incremental approach to conformant planning
  - Operates in an anytime manner
  - Fastest conformant planner on problems with parallelism
  - Described in Kurien, Nayak and Smith, AIPS 2002
- Planning With Failures: SCOPE (in preparation)
  - Novel approach when desired plan is not possible
  - Demonstrates multiple strategies for reducing planning scope

Challenges

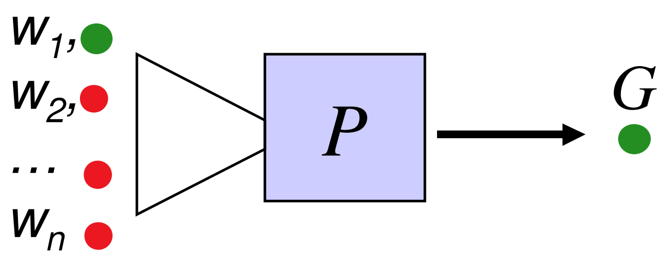


parc Palo Alto Research Center

Conformant Planning

- Problem Instance
  - Let *Domain* be a description of a planning domain
  - Let *Worlds* be a set of initial states of the domain, $\{w_1, w_2, \ldots w_n\}$
  - Let *G* be a goal description
  - There are no sensing actions

- Task: Find plan *P* that applied to any w_i results in a state entailing *G*



- *P* is a conformant plan

- Challenge: Actions chosen in w_i may have undesirable effects in w_j

Existing Approaches to Conformant Planning

- Select actions for *P* by considering all *Worlds* simultaneously

CGP	Smith & Weld 1998	Graphplan over multiple plan graphs
CMBP	Cimatti & Roveri 1999	BDD representation of belief state
GPT	Bonet & Geffner 2001	Heuristic search in space of belief states
HSCP	Bertoli, Cimatti & Roveri 2001	BDD + heuristic search

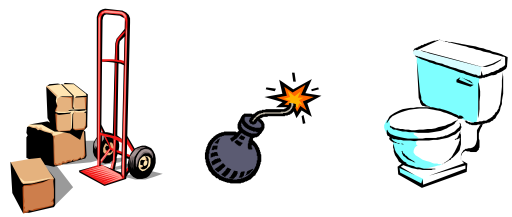
- Generate a plan in w_i and test if it achieves *G* in all *Worlds*

CPlan	Castellini, Giunchiglia & Tachella 2001	SAT encoding determines possible plans which must be checked

parc Palo Alto Research Center

An Observation on Conformant Plans

- Example Domain: Bomb in the Toilet
  - Set of N packages, p1 through pN
  - Packages may have bombs (1, many, a subset)
  - Bombs defused by dunking the package in the toilet
  - The toilet must be flushed before dunking again



- Example Problem
  - 1 toilet
  - 6 packages
  - A bomb is in p1, p2, p3, p5 or (p4 & p6)

Bomb in the Toilet

6 packages, 1 toilet

Plan Step	Action
1	Dunk p3
2	Flush
3	Dunk p2
4	Flush
5	Dunk p1
6	Flush
7	Dunk p6
8	Flush
9	Dunk p4
10	Flush
11	Dunk p5

parc Palo Alto Research Center

An Observation on Conformant Plans

- Example Domain: Bomb in the Toilet
  - Set of N packages, p1 through pN
  - Packages may have bombs (1, many, a subset)
  - Bombs defused by dunking the package in the toilet
  - The toilet must be flushed before dunking again

- Fragment if bomb in p1

Bomb in the Toilet

6 packages, 1 toilet

Plan Step	Action
1	Dunk p3
2	Flush
3	Dunk p2
4	Flush
5	**Dunk p1**
6	Flush
7	Dunk p6
8	Flush
9	Dunk p4
10	Flush
11	Dunk p5

parc Palo Alto Research Center

An Observation on Conformant Plans

- Example Domain: Bomb in the Toilet
  - Set of N packages, p1 through pN
  - Packages may have bombs (1, many, a subset)
  - Bombs defused by dunking the package in the toilet
  - The toilet must be flushed before dunking again

- Fragment if bomb in p1

- Fragment if bombs in p6 and p4

Bomb in the Toilet

6 packages, 1 toilet

Plan Step	Action
1	Dunk p3
2	Flush
3	Dunk p2
4	Flush
5	Dunk p1
6	Flush
7	Dunk p6
8	Flush
9	Dunk p4
10	Flush
11	Dunk p5

parc Palo Alto Research Center

An Observation on Conformant Plans

- Example Domain: Bomb in the Toilet
  - Set of N packages, p1 through pN
  - Packages may have bombs (1, many, a subset)
  - Bombs defused by dunking the package in the toilet
  - The toilet must be flushed before dunking again

- Fragment if bomb in p1 → steps 4–5
- Repair action to unify fragments → step 6
- Fragment if bombs in p6 and p4 → steps 8–9

Bomb in the Toilet

6 packages, 1 toilet

Plan Step	Action
1	Dunk p3
2	Flush
3	Dunk p2
4	Flush
5	Dunk p1
6	Flush
7	Dunk p6
8	Flush
9	Dunk p4
10	Flush
11	Dunk p5

parc Palo Alto Research Center

An Observation on Conformant Plans

- Example Domain: Bomb in the Toilet
  - Set of N packages, p1 through pN
  - Packages may have bombs (1, many, a subset)
  - Bombs defused by dunking the package in the toilet
  - The toilet must be flushed before dunking again

- Every conformant plan P must contain a fragment that achieves the goal in each world

- Each world has plans that are fragments of some P

- Approach:

  Grow a set of fragments into a conformant plan

Bomb in the Toilet

6 packages, 1 toilet

Plan Step	Action
1	Dunk p3
2	Flush
3	Dunk p2
4	Flush
5	Dunk p1
6	Flush
7	Dunk p6
8	Flush
9	Dunk p4
10	Flush
11	Dunk p5

parc Palo Alto Research Center

Fragment-based Conformant Planning

- Intuition

For each w_i in *Worlds* {

1. Generate a plan for *Domain* to achieve G in w_i
2. Add the planned actions to *Domain*

}

- Step 2 ensures the plan for w_{i+1} includes the actions that achieved G in $\{w_1 \ldots w_i\}$

parc Palo Alto Research Center

Fragment-based Conformant Planning

Planning Process

Plan Step	Plan for p1					
1	Dunk p1					
2						
3						
4						
5						

parc
Palo Alto Research Center

Fragment-based Conformant Planning

Planning Process

Plan Step	Plan for p1	Fragments for p2 plan				
1	Dunk p1	Dunk p1				
2						
3						
4						
5						

parc Palo Alto Research Center

Fragment-based Conformant Planning

Planning Process

Plan Step	Plan for p1	Fragments for p2 plan	Plan for {p1,p2}			
1	Dunk p1	Dunk p1	Dunk p1			
2						
3			Flush			
4						
5			Dunk p2			

Fragment-based Conformant Planning

Planning Process

Plan Step	Plan for p1	Fragments for p2 plan	Plan for {p1,p2}	Extracted fragment		
1	Dunk p1	Dunk p1	Dunk p1			
2						
3			Flush			
4						
5			Dunk p2	Dunk p2		

parc Palo Alto Research Center

Fragment-based Conformant Planning

Planning Process

Plan Step	Plan for p1	Fragments for p2 plan	Plan for {p1,p2}	Extracted fragment	Fragments for p3 plan	
1	Dunk p1	Dunk p1	Dunk p1		Dunk p1	
2						
3			Flush			
4						
5			Dunk p2	Dunk p2	Dunk p2	

parc Palo Alto Research Center

Fragment-based Conformant Planning

Planning Process

Plan Step	Plan for p1	Fragments for p2 plan	Plan for {p1,p2}	Extracted fragment	Fragments for p3 plan	Plan for {p1,p2,p3}
1	Dunk p1	Dunk p1	Dunk p1		Dunk p1	Dunk p1
2						Flush
3			Flush			Dunk p3
4						Flush
5			Dunk p2	Dunk p2	Dunk p2	Dunk p2

- Search will be required
  - The fragment chosen for w1 may not allow a plan for w2
  - The fragment chosen for w2 may disrupt the plan for w1

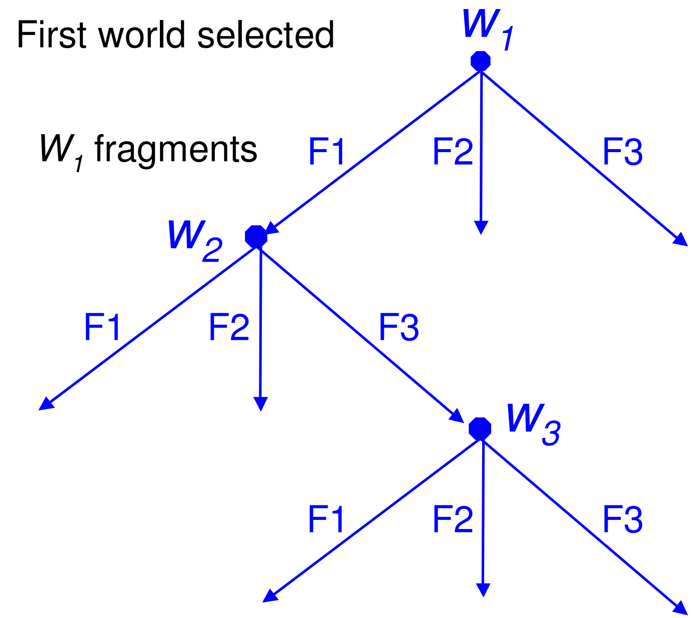
parc Palo Alto Research Center

The FragPlan Algorithm

completed=$\emptyset$
While (*Worlds* $\neq \emptyset$) {
select and remove world w_i from *Worlds*
Choose a plan P_i for *Domain* that achieves G in w_i
Fail if P_i doesn't achieve G for all $w \in$ *completed*
Extract fragment F_i from P_i
Domain = *Domain* + F_i
add w_i to *completed* }
Return P_i

parc Palo Alto Research Center

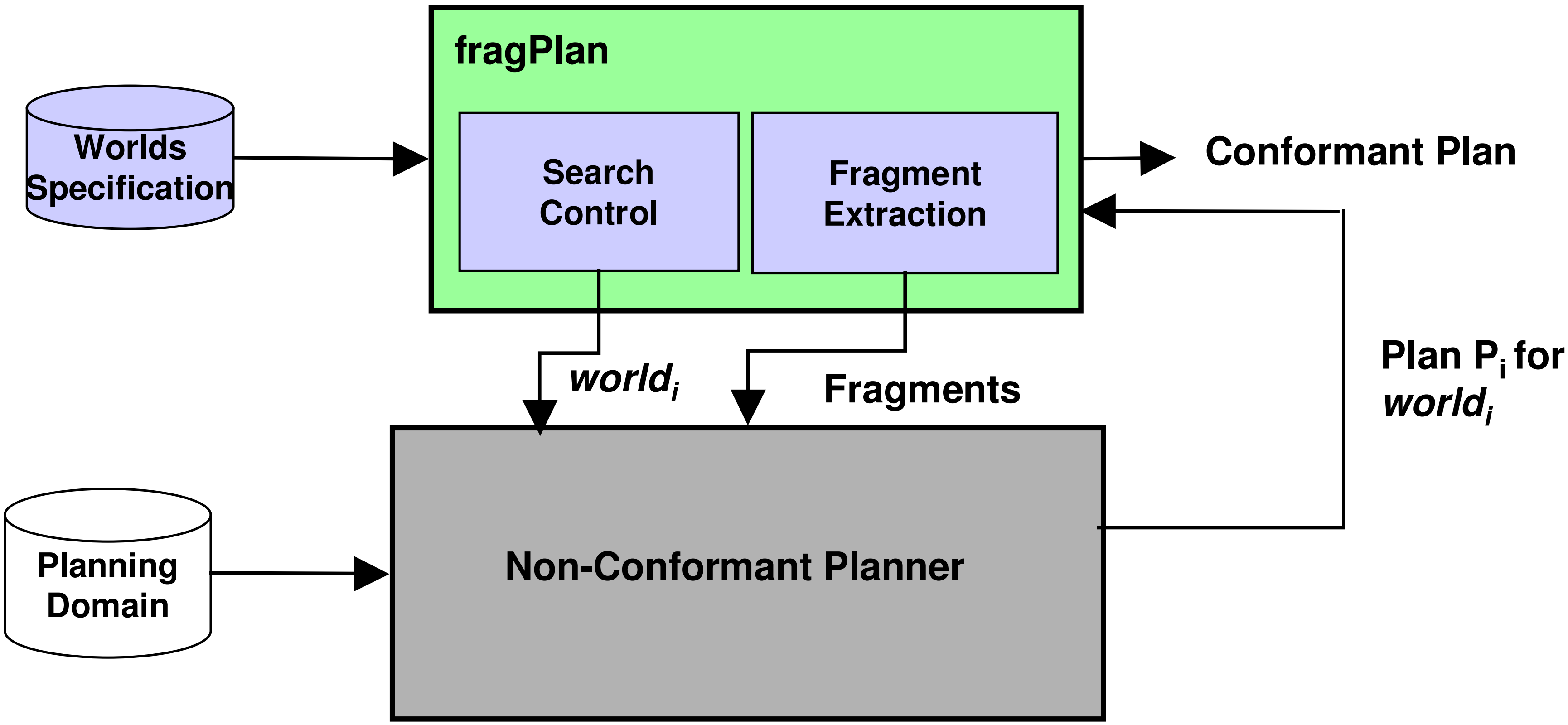
Search Strategies



- Chronological Backtracking
- Probing
  - Extend fragments to as many worlds as possible, then restart
  - On failure, discard all fragments and empty *completed*
  - Effective even when a small subset of worlds are very difficult
  - Fits well with deterministic planner we use to choose P_i for w_i
- Bubbling
  - Find difficult worlds. Solve first by moving them up the stack.

parc Palo Alto Research Center

Implementation



- fragPlan uses a non-conformant planner as a black box
- We need only to be able to force the planner to include fragments

Domains Considered

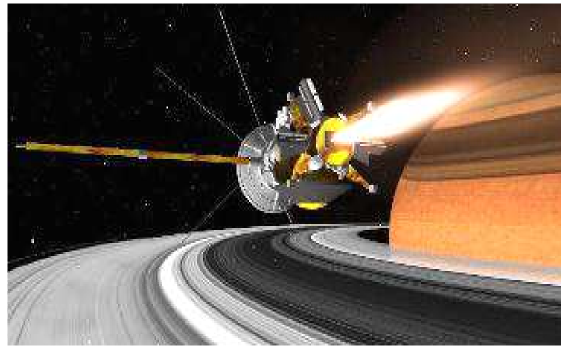
- Hydraulic/Electric Networks
- Grid Worlds
  - A robot is in a ring of rooms
  - It must close and lock all windows
- Logistics
  - Packages must be delivered to various cities
  - Some roads have mines
- Bomb in the Toilet
  - A set of packages arrive, one or more has a bomb
  - Detailed results available for many planners
- Focused on problems solvable on a reactive (few seconds) scale
- World sets tested up to size 150, goal sets up to size 10

Unique Characteristics of fragPlan

- Novel algorithm for conformant planning that performs well on both serial and parallel problems
- Constructive approach
  - We always have a plan, improves in an anytime manner
  - Can delete and add worlds and re-use partial results
- More scalable than other possible worlds approaches
  - Memory usage is constant as the number of worlds increases
  - Computation is less susceptible to explosive growth

parc Palo Alto Research Center

Is Conformant Planning Enough?



- Typical Goals
  - Configure the spacecraft to thrust to enter orbit
  - Configure the camera to take science images on approach

- Typical Safety Constraints
  - Turn the amp off before switching transponders to avoid burn out.
  - Once a device is on, never turn it off. It might not come back on.
  - I’m loathe to blow the pyro valves that enable the backup engine

- Typical Failures
  - The camera is dead, it’s power popped off, or its interface is hung
  - Thruster +x+y or –x-y is clogged

parc
Palo Alto Research Center

SCOPE - Safe, Conformant, Optimizing Planning Engine

- Goal: Find the best possible plan in the available time
- Approach: Manipulate the scope of the problem

```
While (Time ≠ 0) {
    select constraints from {Worlds ∪ Goals ∪ Safety}
    FragPlan(constraints)
    }
```

- Challenge:
  – Which subsets of {*Worlds* $\cup$ *G* $\cup$ *S*} admit a plan?
  – Will we have a plan when time runs out?

parc Palo Alto Research Center

SCOPE – Safe, Conformant, Optimizing Planning Engine

- Approach: Manipulate the scope of the problem

While ($Time \neq 0$) {

select constraints from {$Worlds \cup Goals \cup Safety$}

FragPlan(constraints) for some time

}

Balance solving current constraints vs. exploration

parc Palo Alto Research Center

SCOPE – Safe, Conformant, Optimizing Planning Engine

- Approach: Manipulate the scope of the problem

While ($Time \neq 0$) {
    select constraints from {$Worlds \cup Goals \cup Safety$}
    FragPlan(constraints) for some time
    }

- Pareto-optimality requires checking all constraint subsets

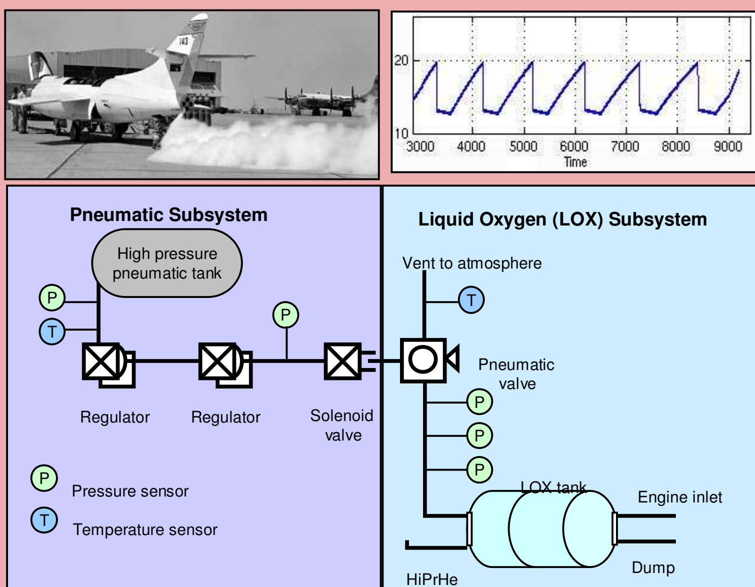
- We have developed many simpler selection policies and experimented with several

Status

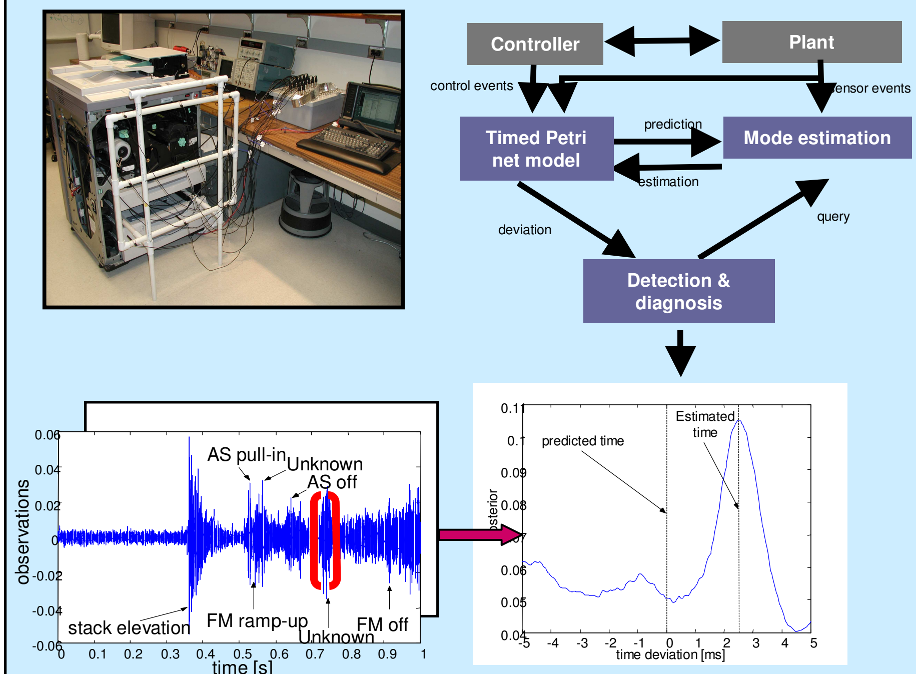
- Diagnosis or State Estimation: L2 (Kurien & Nayak, AAAI 200)
  - Tracks multiple system trajectories
  - Backtracks to find failures that were not immediately observable
  - Extends ideas of Livingstone (Williams and Nayak, AAAI 1996) as flown on DS1
  - Used by S/C engineers to develop X-34, X-37 models & diagnostic scenarios
  - In use at NASA, licensed to a spacecraft software company
- Conformant Planning: fragPlan (Kurien, Nayak & Smith, AIPS 2002)
  - Novel, incremental approach to conformant planning
  - Operates in an anytime manner
  - Fastest conformant planner on problems with parallelism
  - Described in Kurien, Nayak and Smith, AIPS 2002
- Planning With Failures: SCOPE (in preparation)
  - Novel approach when desired plan is not possible
  - Demonstrates multiple strategies for reducing planning scope

Some Current Work at PARC

Hybrid and Distributed Diagnosis



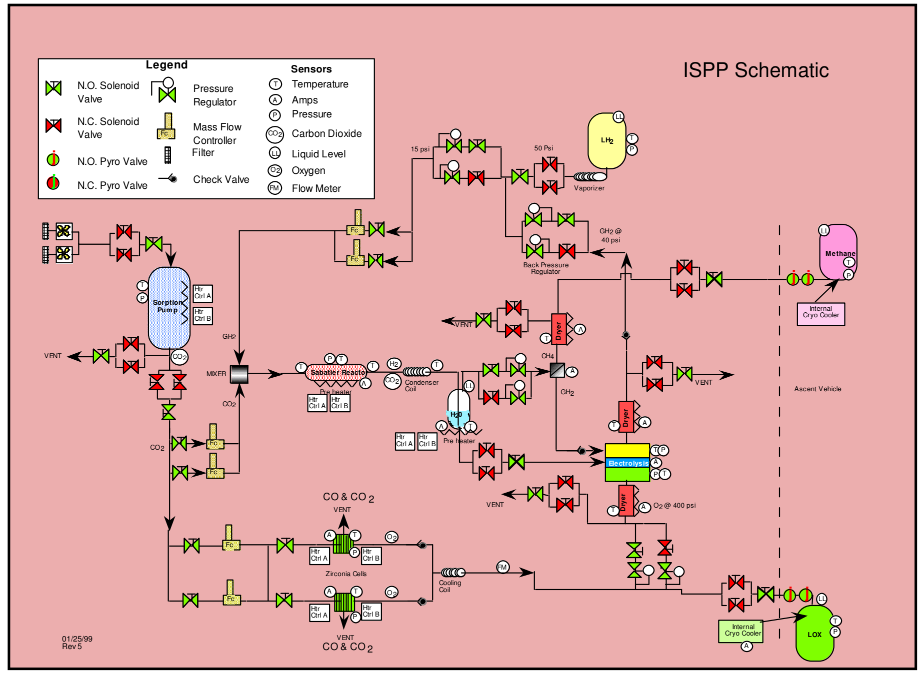
Diagnosis of Continuous Systems



Backing Slides

parc
Palo Alto Research Center

Cool Problems at NASA & PARC



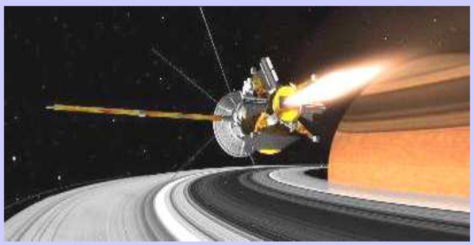
- How do we make complex systems autonomous?
- How can they continue operating after failures?

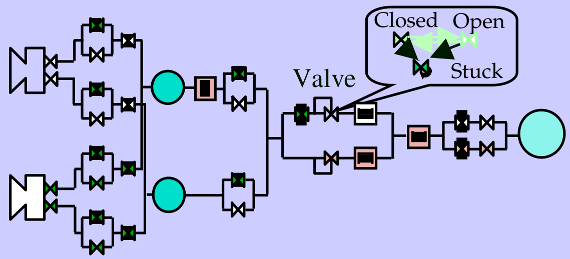
parc Palo Alto Research Center

What is planning?

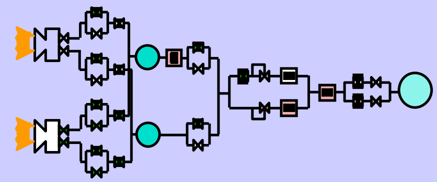
Typical Application	Domain Model	Typical Goal

Autonomous Machines

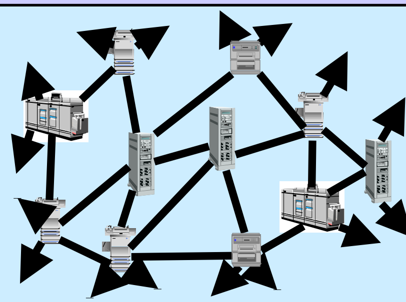




Configure the craft to thrust

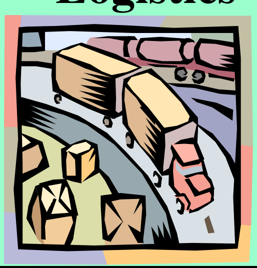


Web Software Agents



Buy me a cheap ticket to Rio during Carnival and print my itinerary at a printer near my office.

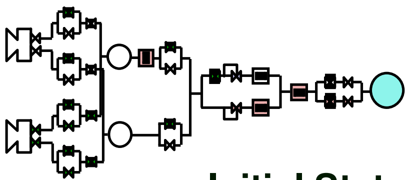
Logistics



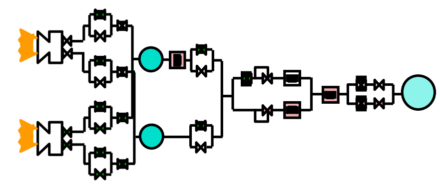
Deliver package A to San Jose, package B to Oakland, package C to Daly City ….

Choosing Actions

Action: Ignite Engine	
Pre:	Oxygen Flowing
	Fuel Flowing
Post:	Engine Thrusting



Initial State

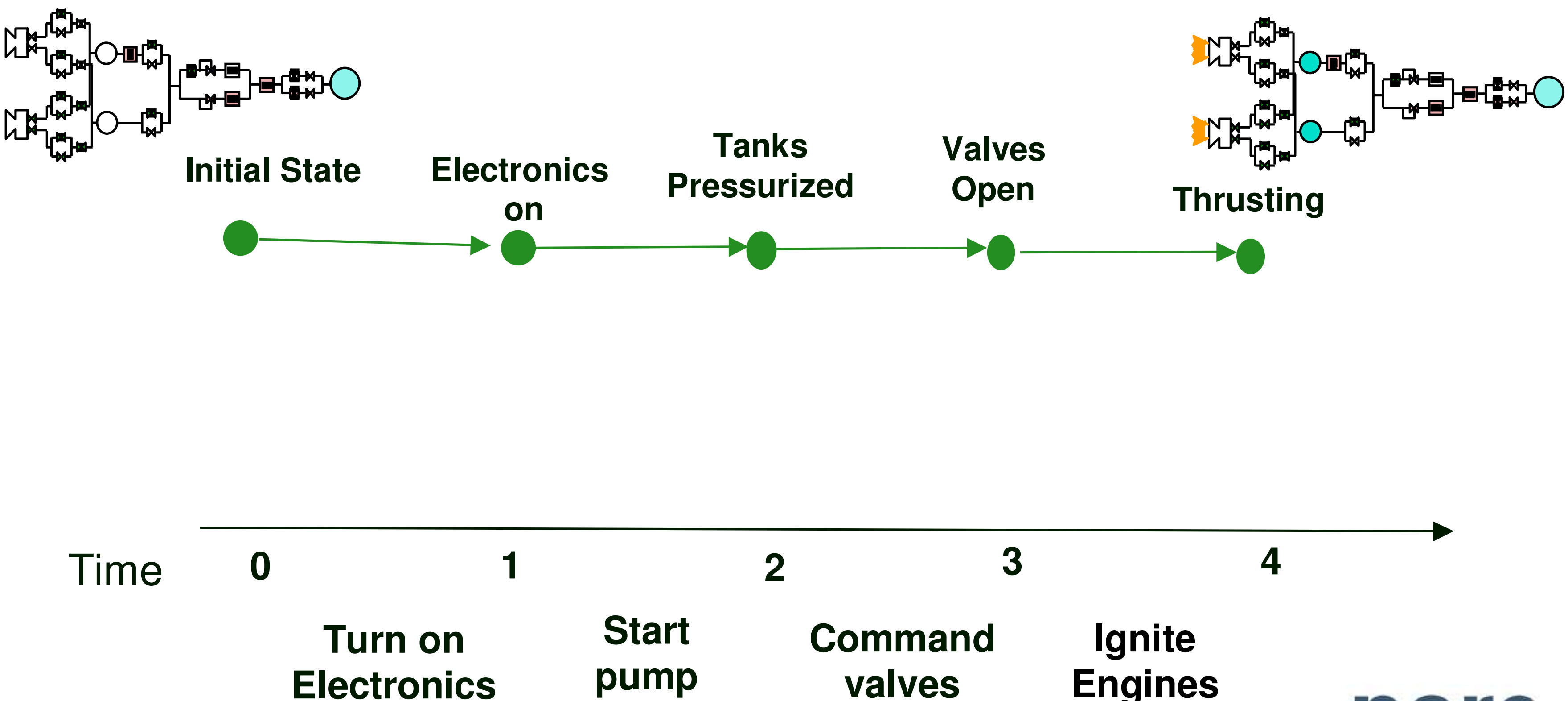


Thrusting

Time 0 1 2 3 4

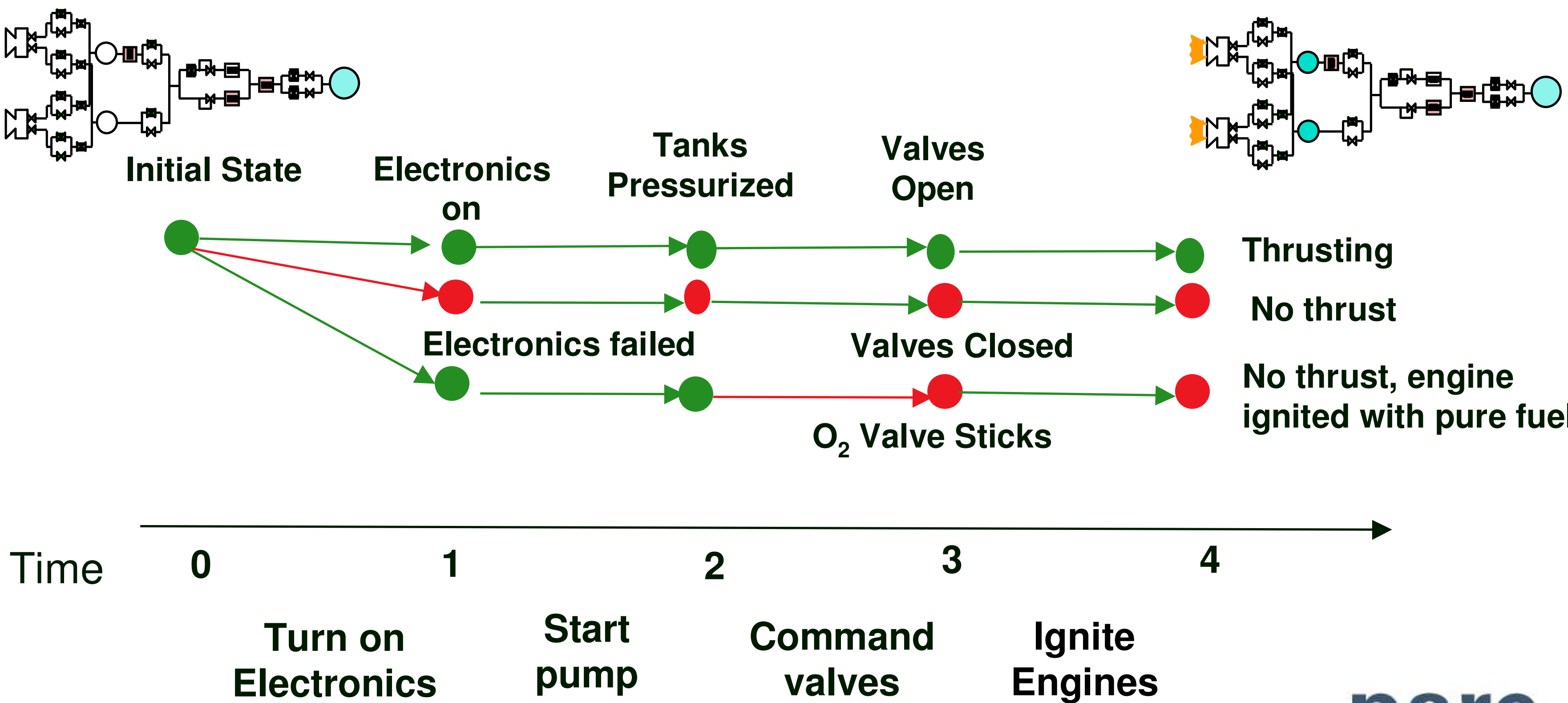
Choosing Actions

Action: Ignite Engine	
Pre:	Oxygen Flowing
	Fuel Flowing
Post:	Engine Thrusting



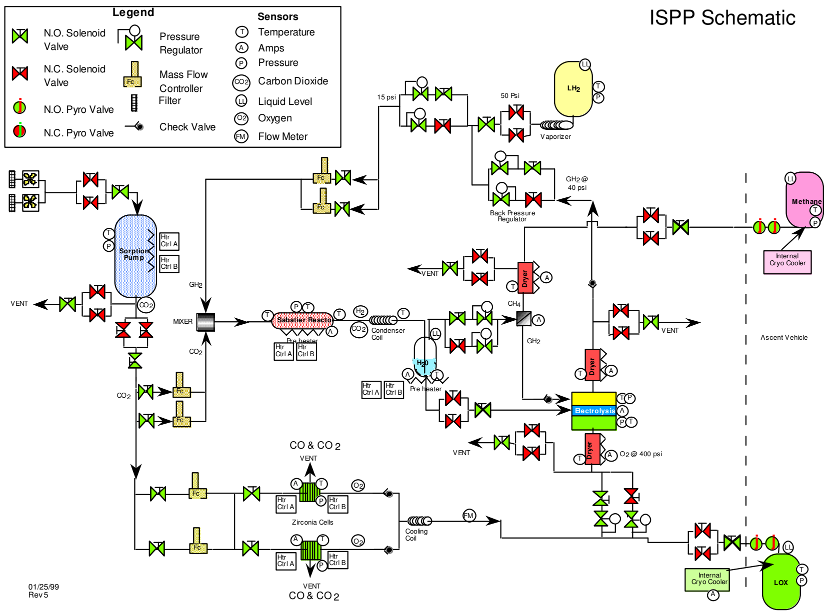
Choosing Actions

Action: Ignite Engine
Pre: Oxygen Flowing
Fuel Flowing
Post: Engine Thrusting

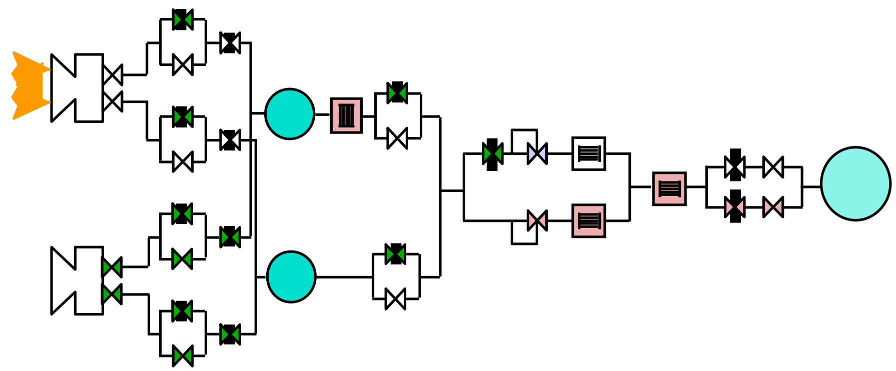


Our State Estimation Problems

Mars Propellant Production Model



Spacecraft Propulsion System Model



- Hundreds of variables
- Typically 10^{150} discrete states
- Mostly deterministic, but components will fail
- Failure probabilities known only by rank or order

parc Palo Alto Research Center

Approaches to State Estimation

- Exact methods
  - Problem: State space is enormous and discontinuous

Kalman filter	Kalman 1960	Continuous only, unimodal, white noise
Dynamic Bayes' net	Pearl 1988	Need to compute huge joint distributions

- Approximate the distribution over the state space
  - Problem: System is almost deterministic with abrupt failures

Approximate DBN	Boyen & Kohler 1998	Depends upon stochasticity assumptions
Particle Filters	Dearden 2002	Particles attracted to likely states

parc
Palo Alto Research Center

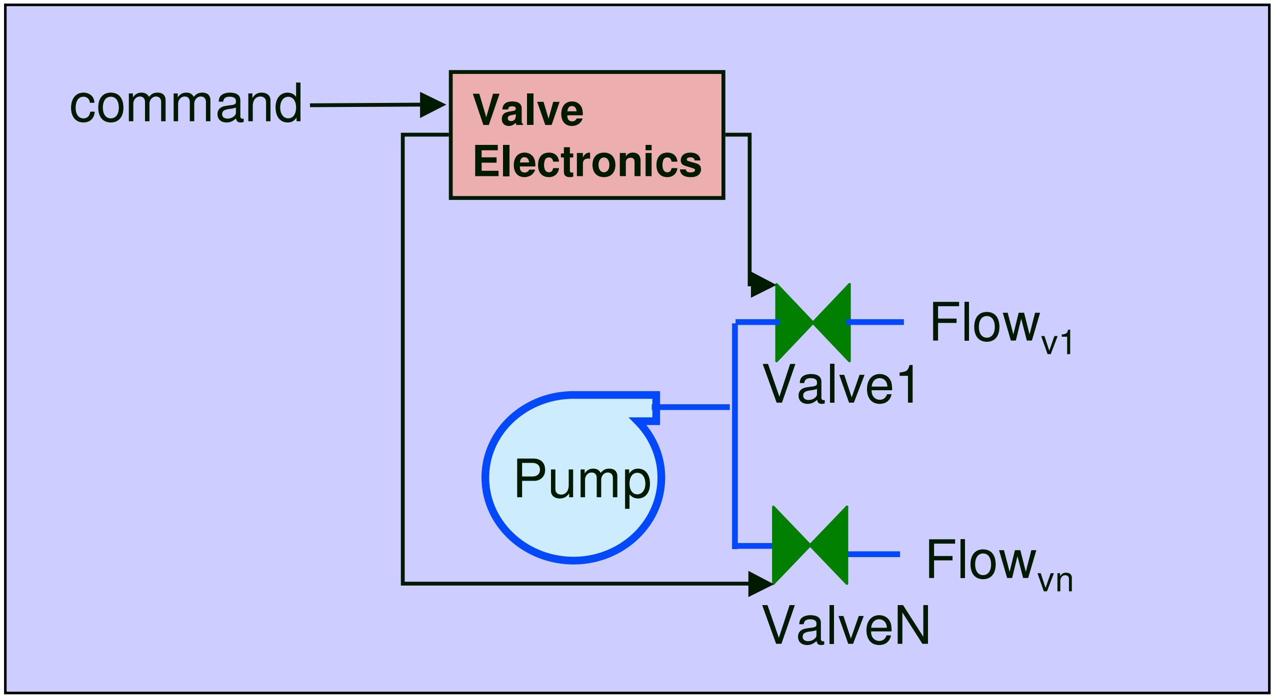
Approaches to State Estimation

- Model-based diagnosis based upon logical consistency
  - Advantage: Compositional
    - Refuting a diagnosis of a component may refute an exponential number of system diagnoses (states)
  - Advantage: Incremental
    - Diagnoses (states) are generated in order of likelihood
  - Problem: Actions or evolution over time not handled well, or at all

Sherlock, GDE	De Kleer & Williams 1989	No actions or state evolution
Livingstone	Williams & Nayak 1996	State evolution, but arbitrarily bad approximation of the most likely state

parc Palo Alto Research Center

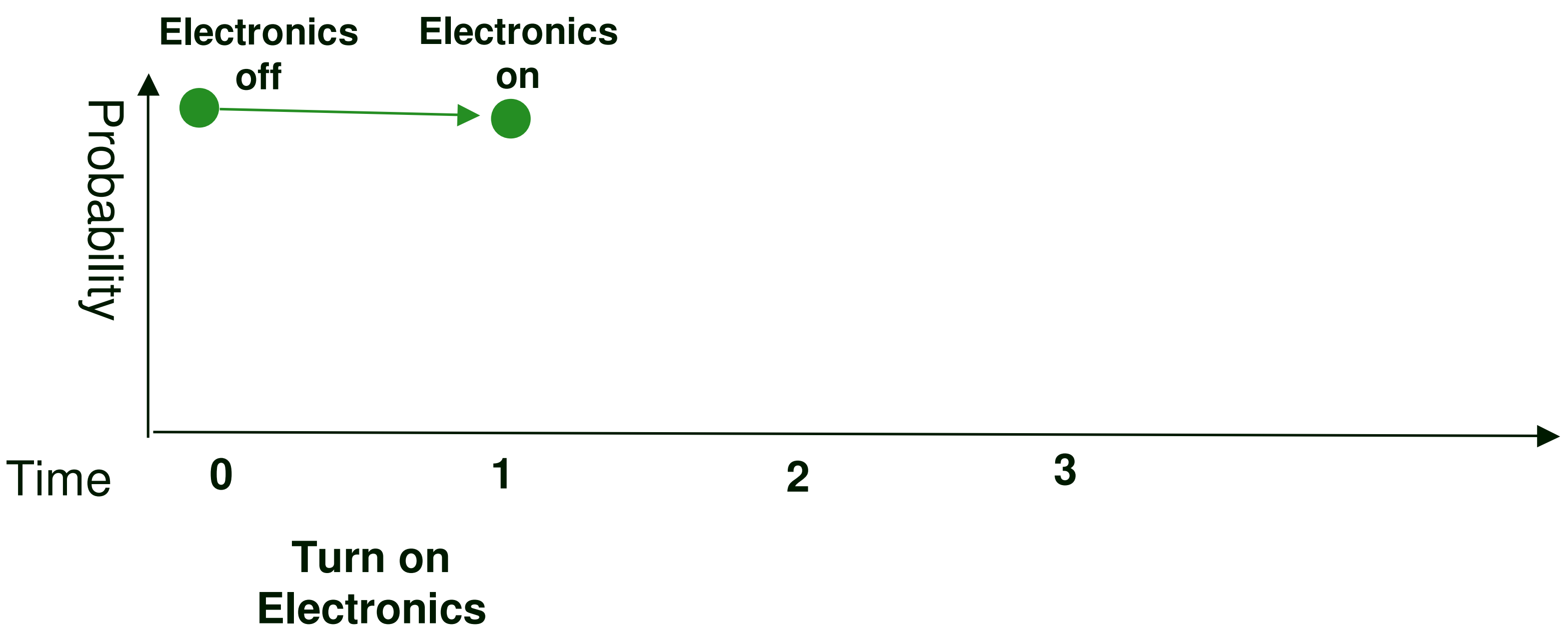
Simple Valve Example



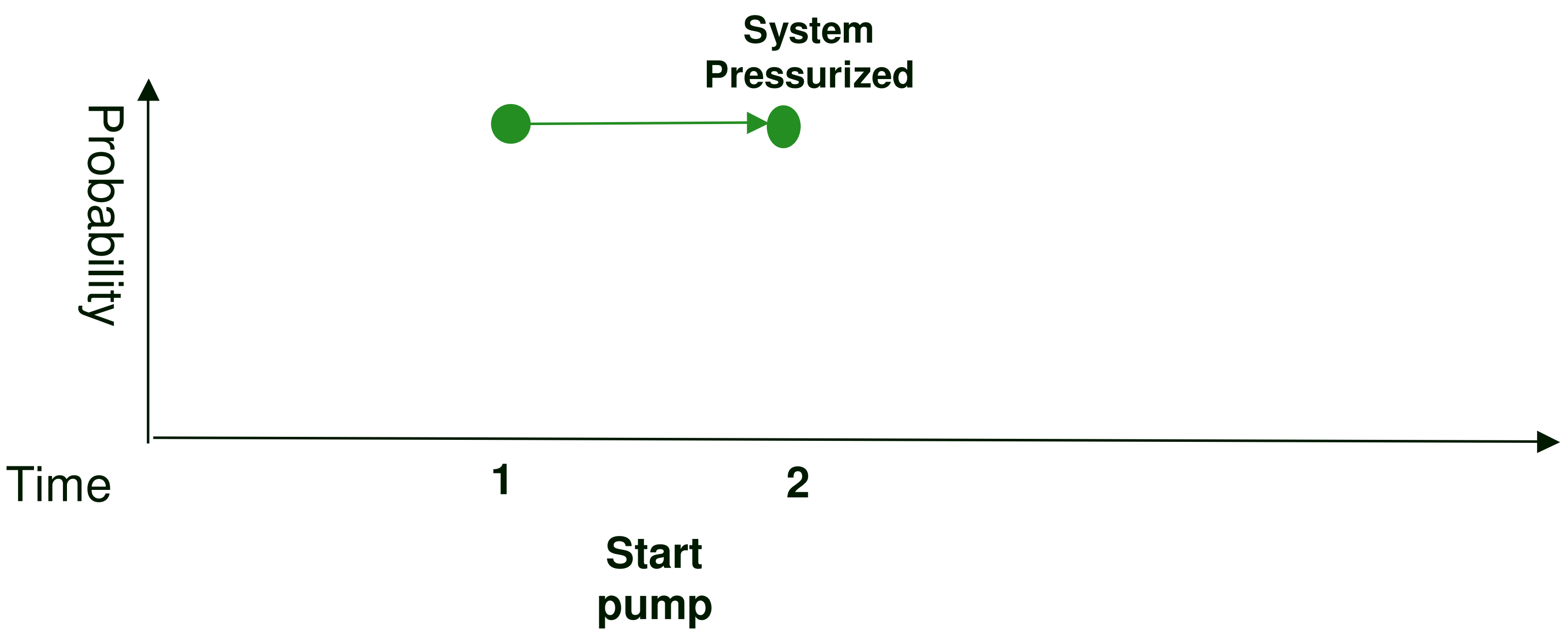
- The pump pressurizes the valves
- Valve electronics send commands to valves
- Flow measured at each valve
- Electronics may hang, valves may stick shut

parc Palo Alto Research Center

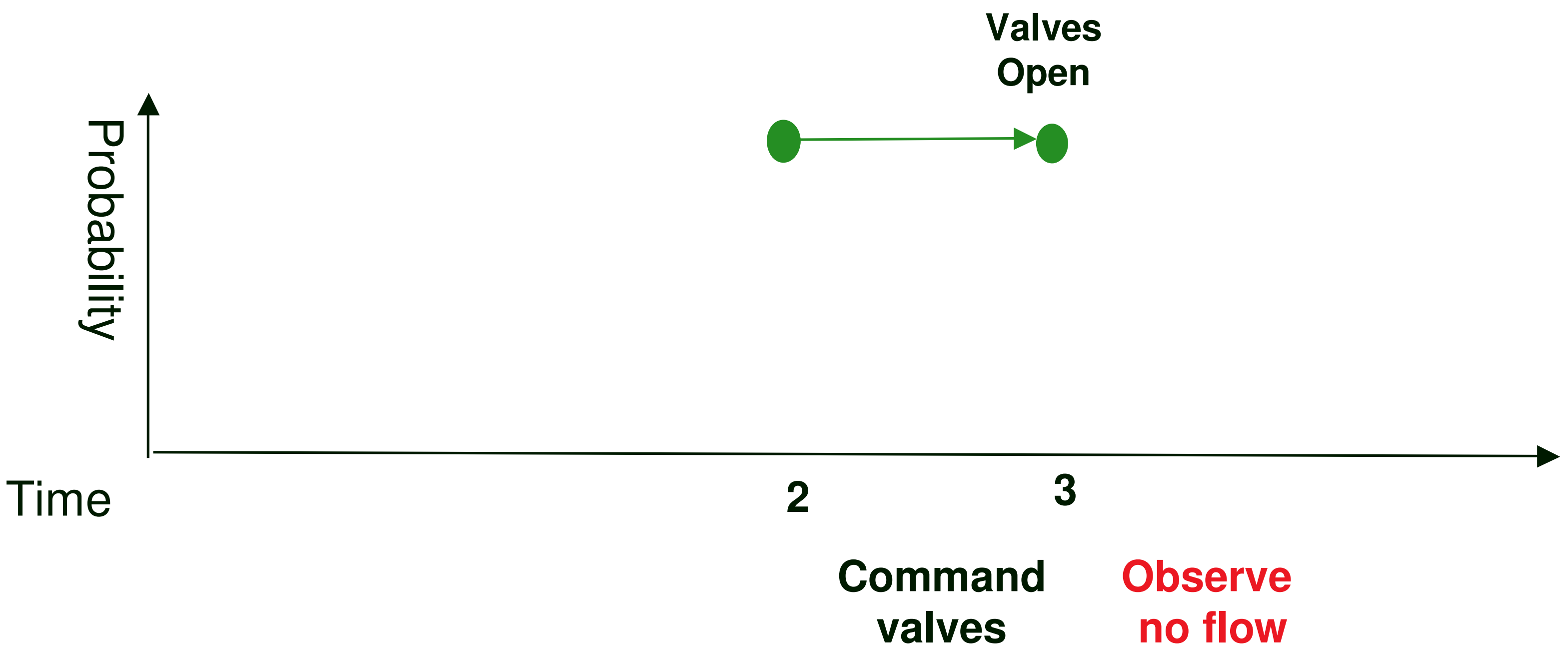
Traditional Diagnosis



Traditional Diagnosis

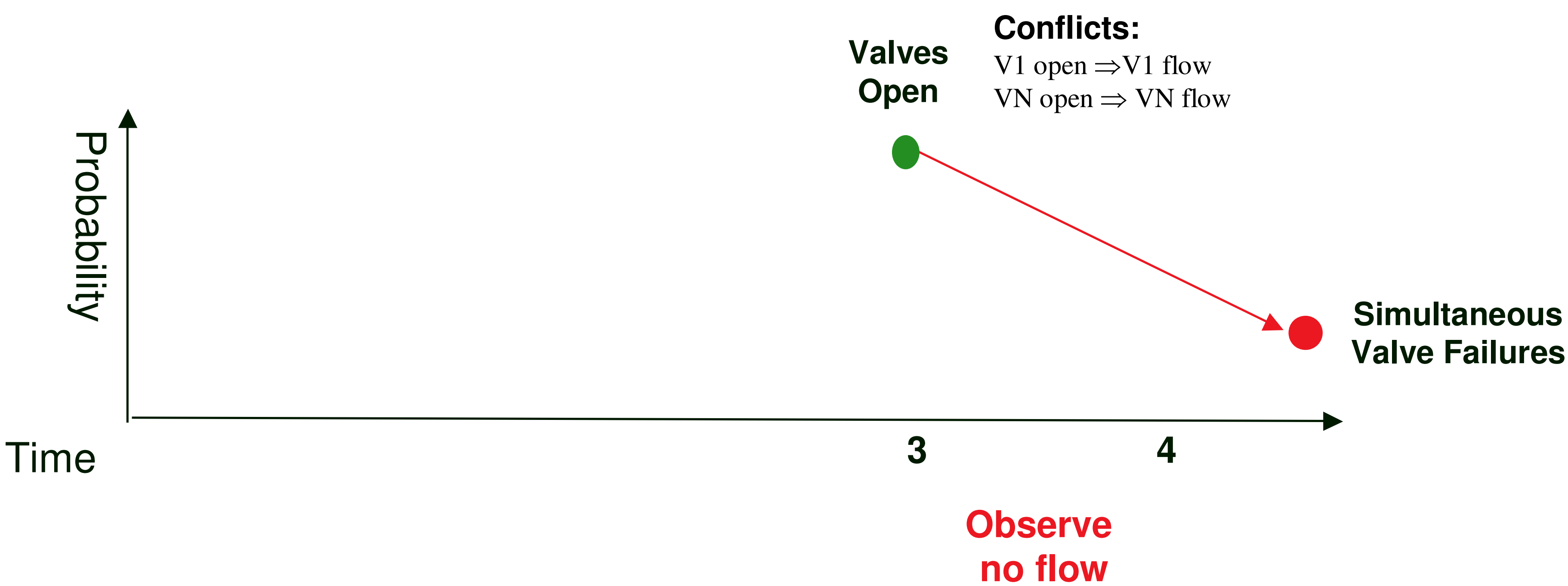


Traditional Diagnosis

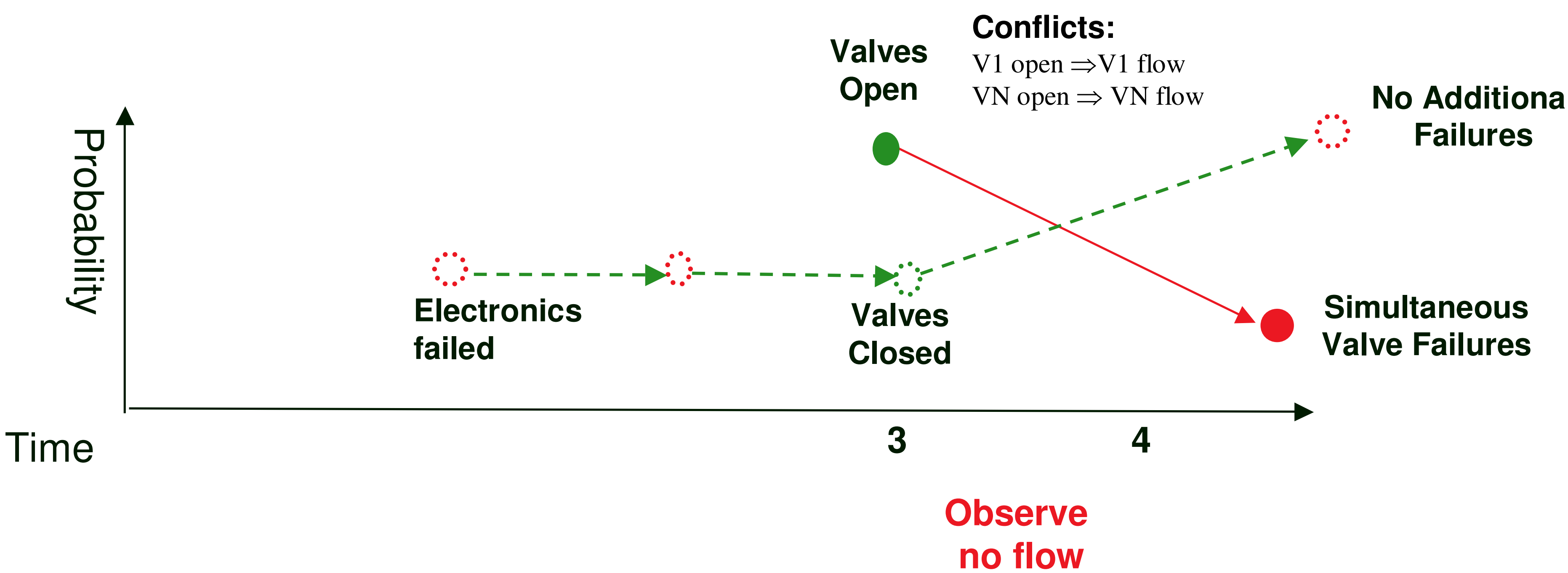


parc
Palo Alto Research Center

Traditional Diagnosis

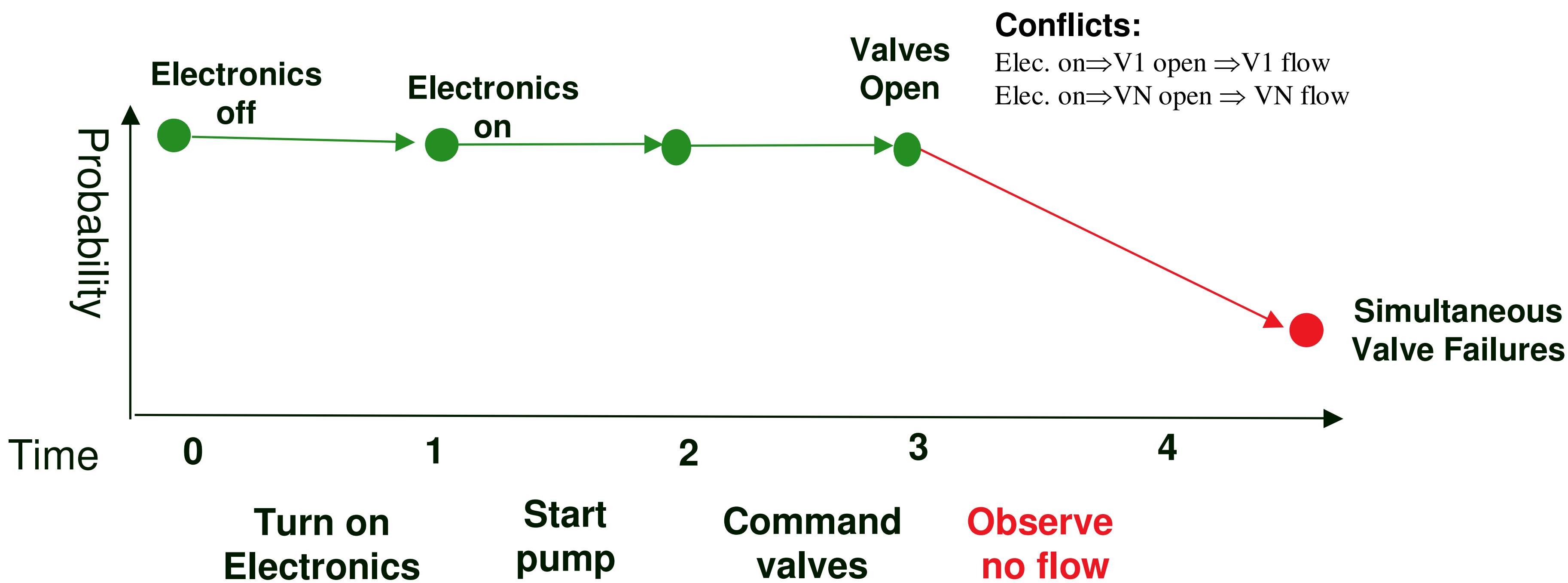


The Problem

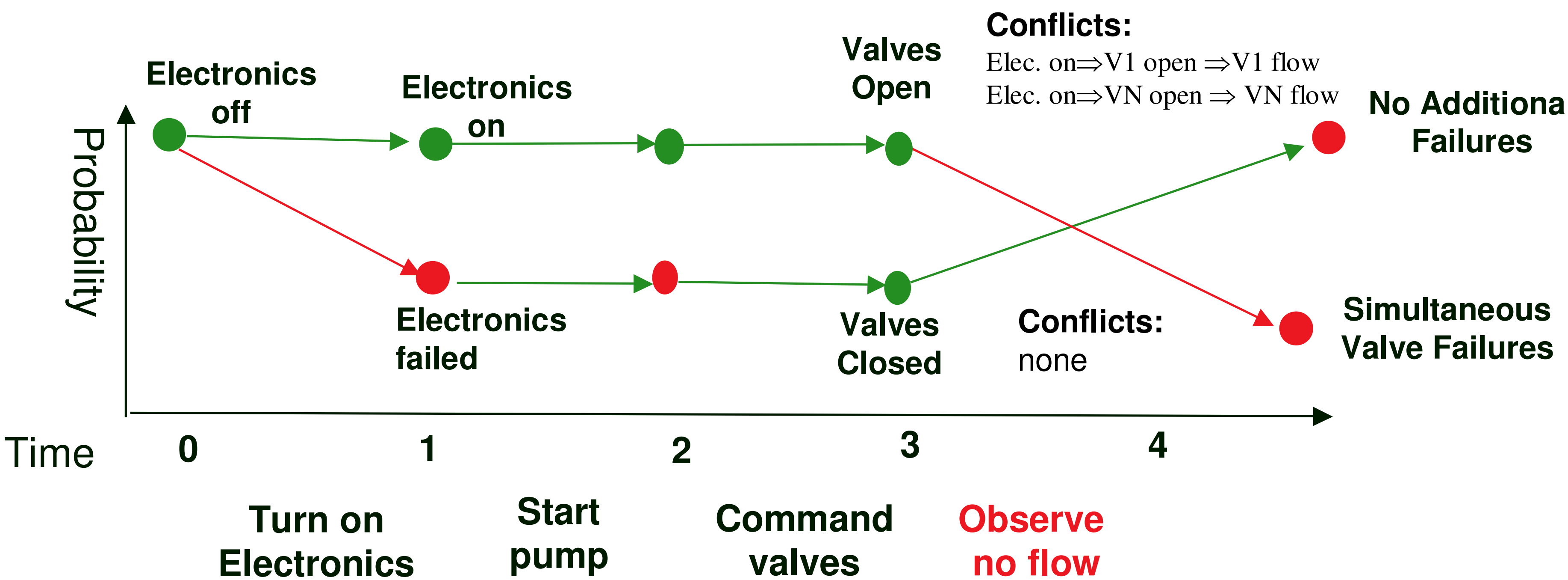


parc Palo Alto Research Center

Generating Trajectories Incrementally



Generating Trajectories Incrementally



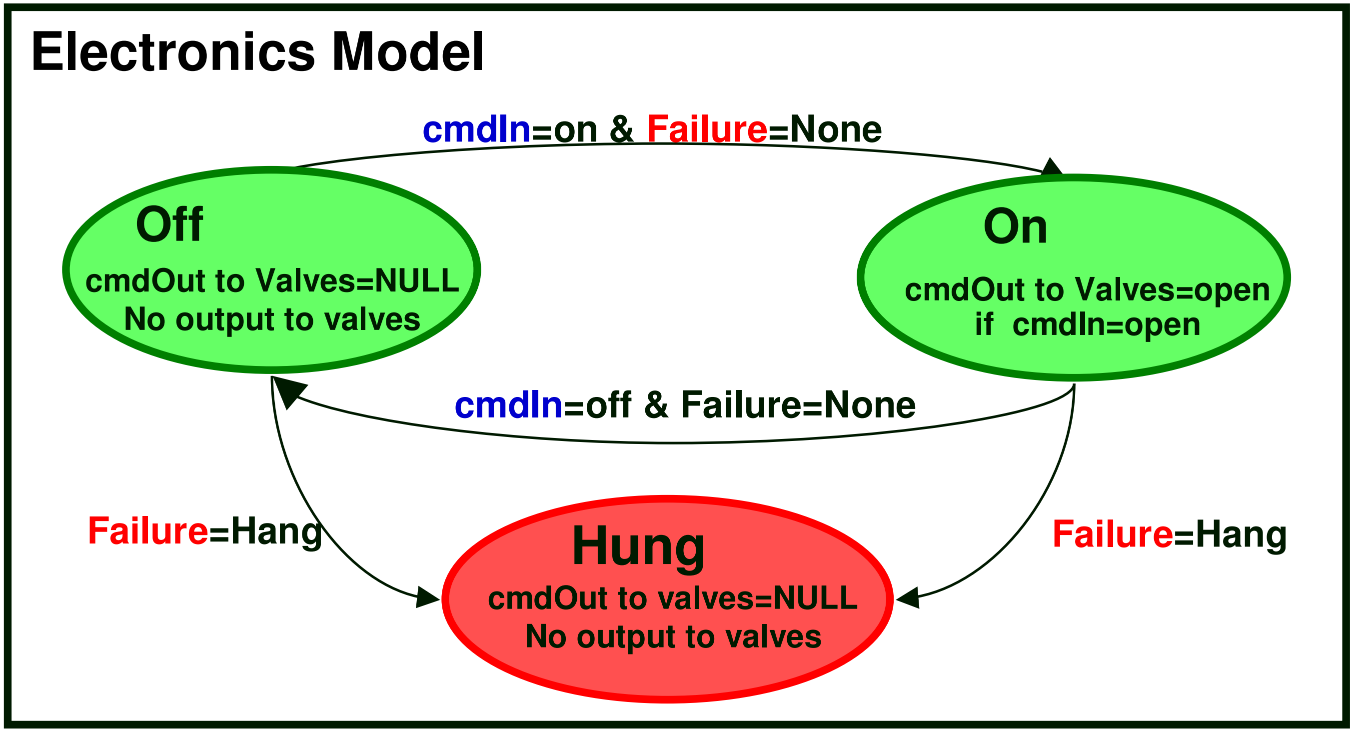
parc
Palo Alto Research Center

Approach

- Create a structure that can enumerate every possible trajectory of the system
- Enumerate N trajectories that are consistent with observations thus far
- Extend each trajectory as actions are taken
- When trajectories are knocked out by new observations, incrementally generate the next most likely trajectory

parc Palo Alto Research Center

Encoding Device Behavior

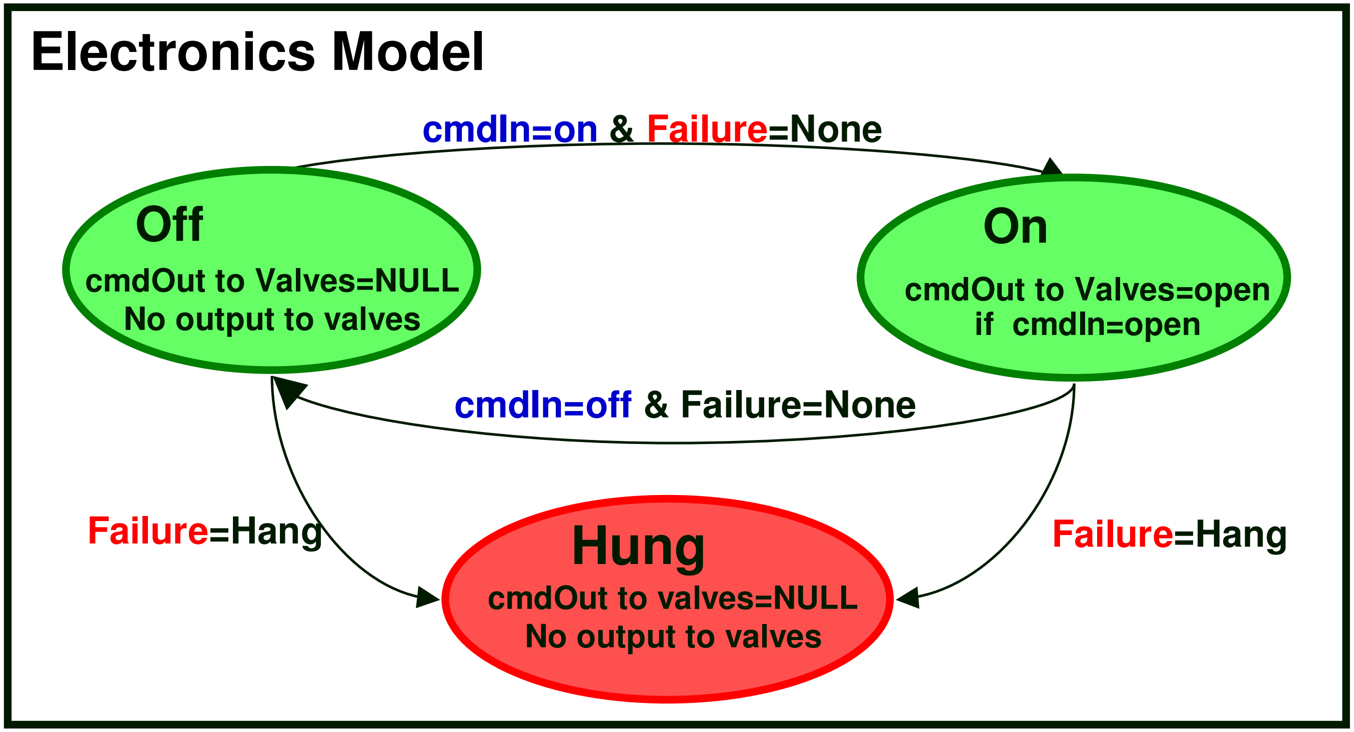


Prior Probabilities

Value	P(**Failure**=Value)
None	α
Hang	$1-\alpha$

parc Palo Alto Research Center

Encoding Device Behavior



Constraint Representation

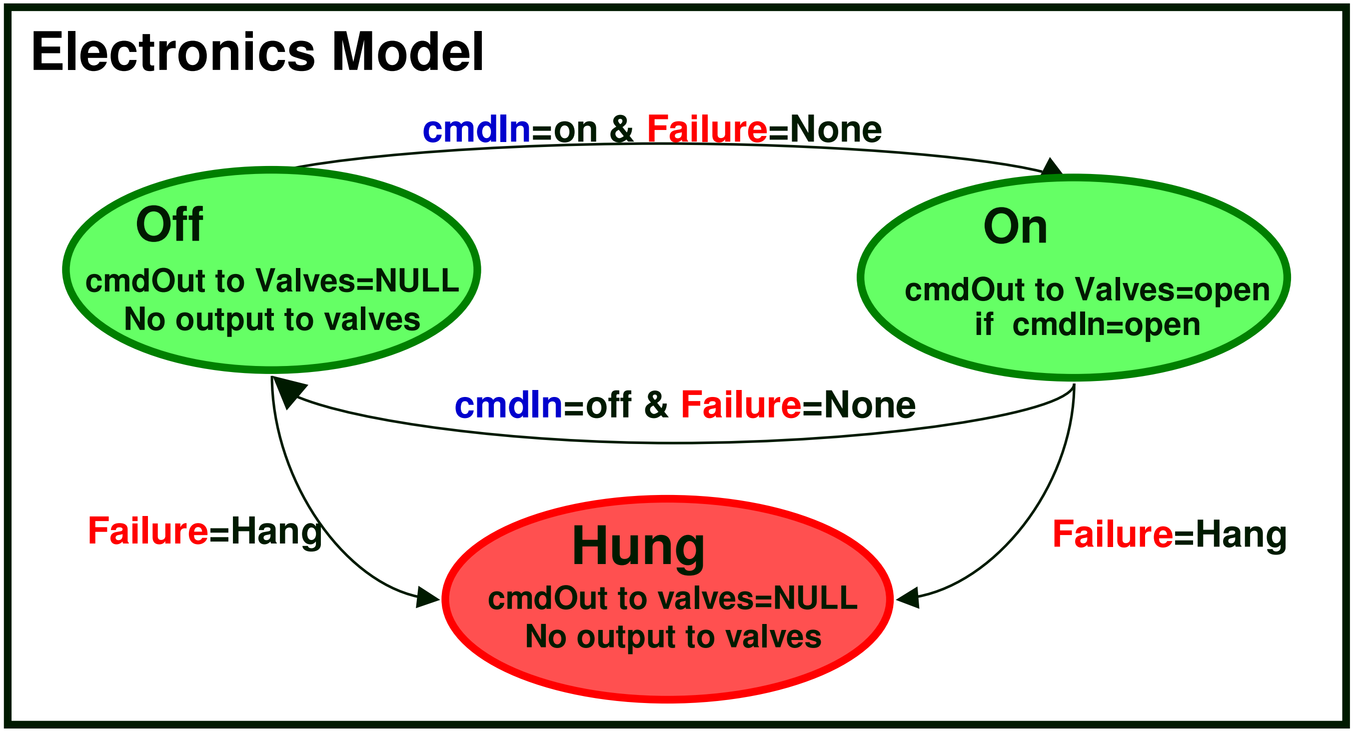
Mode Behavior

mode=Off ⇒ cmdOut=NULL

Transitions	Time t	Time t+1
mode=Off & cmdIn=off & Failure=None ⇒		mode=Off
	Failure=Hang ⇒	mode=Hung

parc Palo Alto Research Center

Encoding Device Behavior

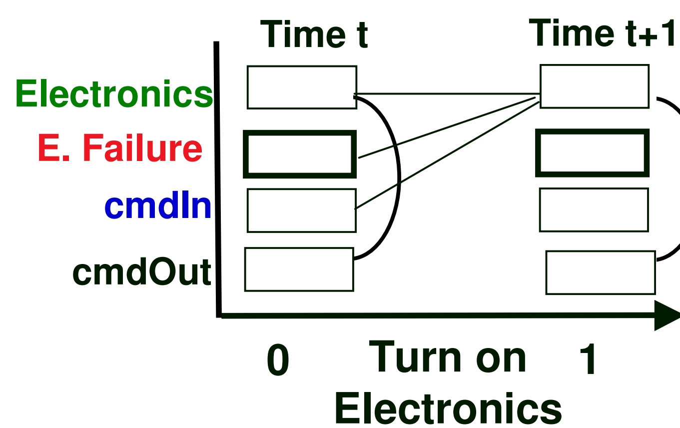


Constraint Representation

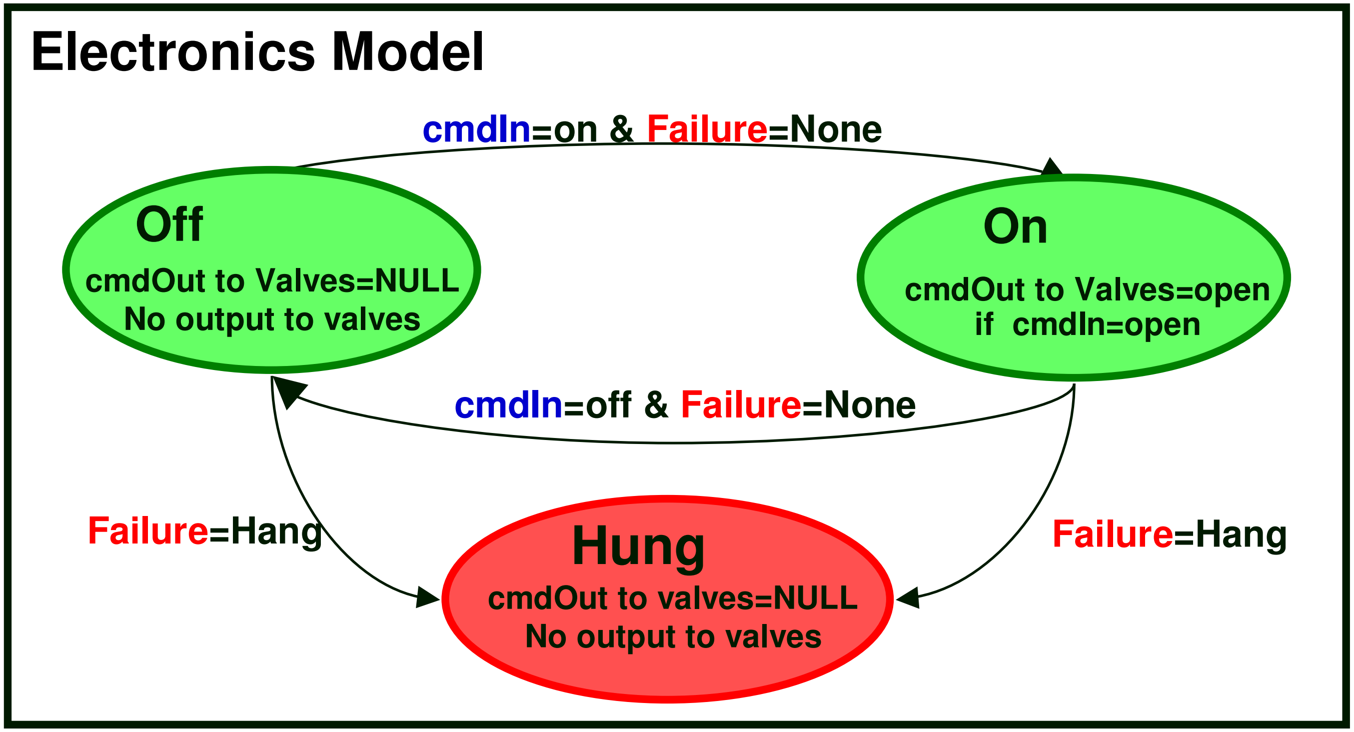
Mode Behavior

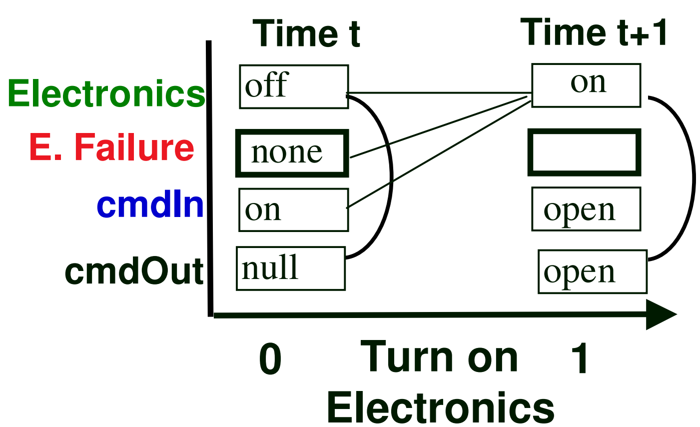
mode=Off $\Rightarrow$ cmdOut=NULL

Transitions	Time t	Time t+1
	mode=Off & cmdIn=off & Failure=None $\Rightarrow$	mode=Off
	Failure=Hang $\Rightarrow$	mode=Hung

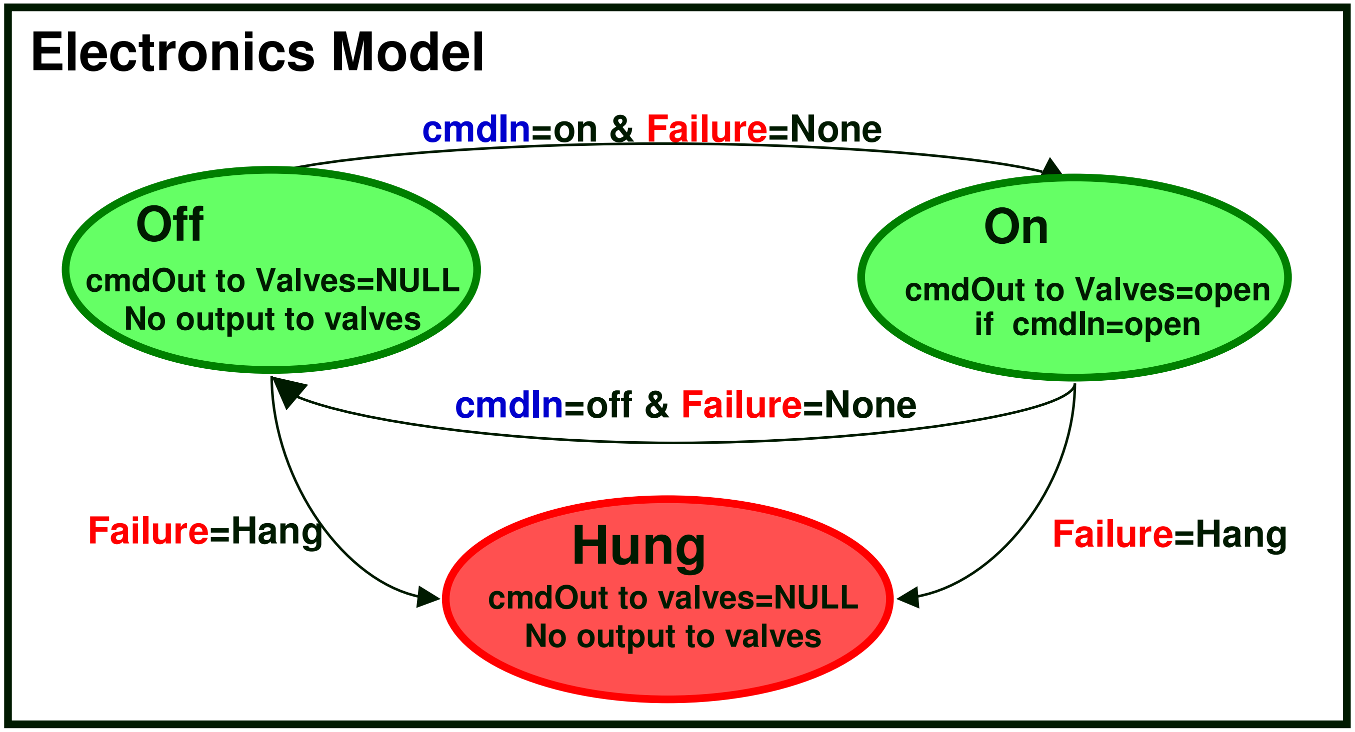


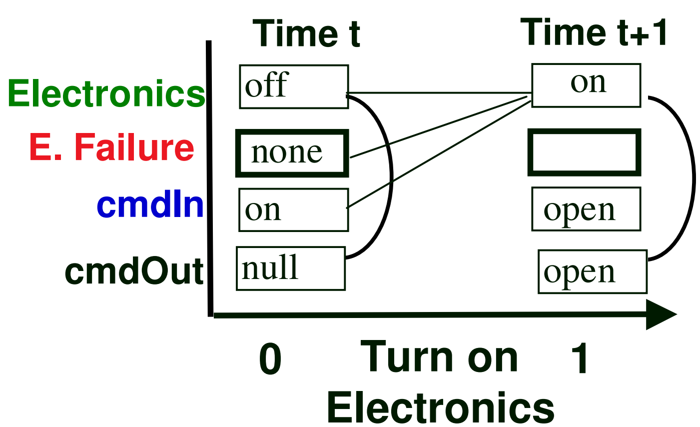
Encoding Device Behavior

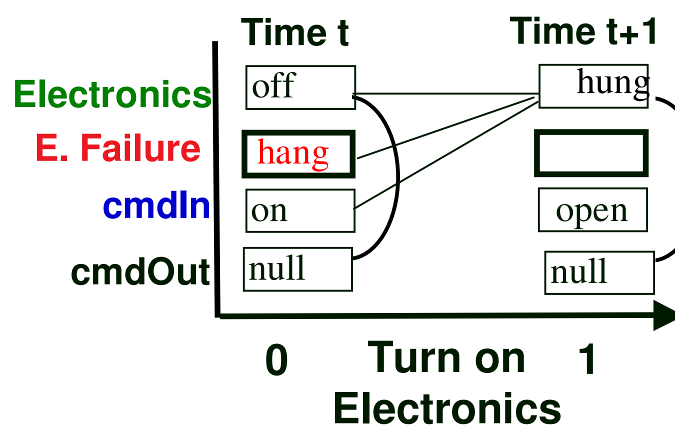




Encoding Device Behavior

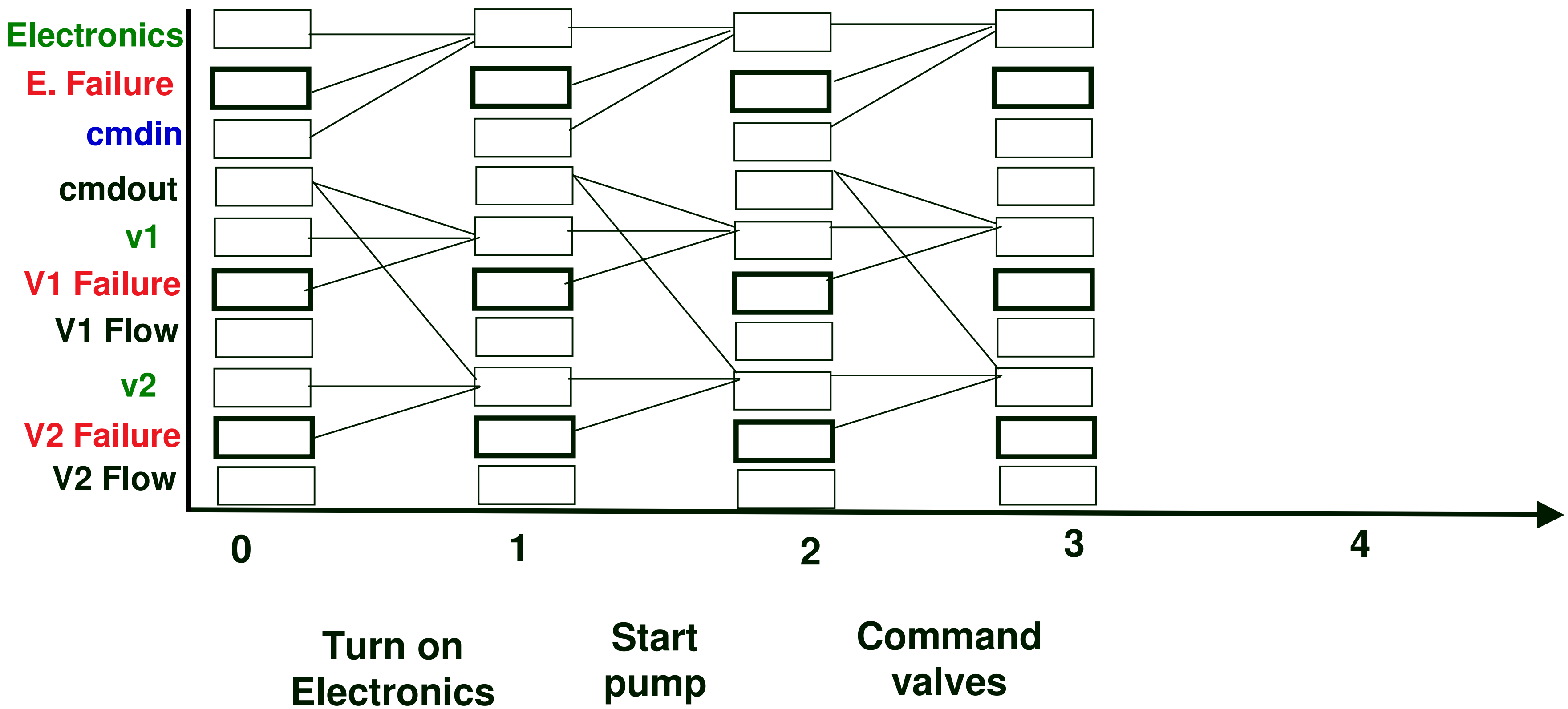






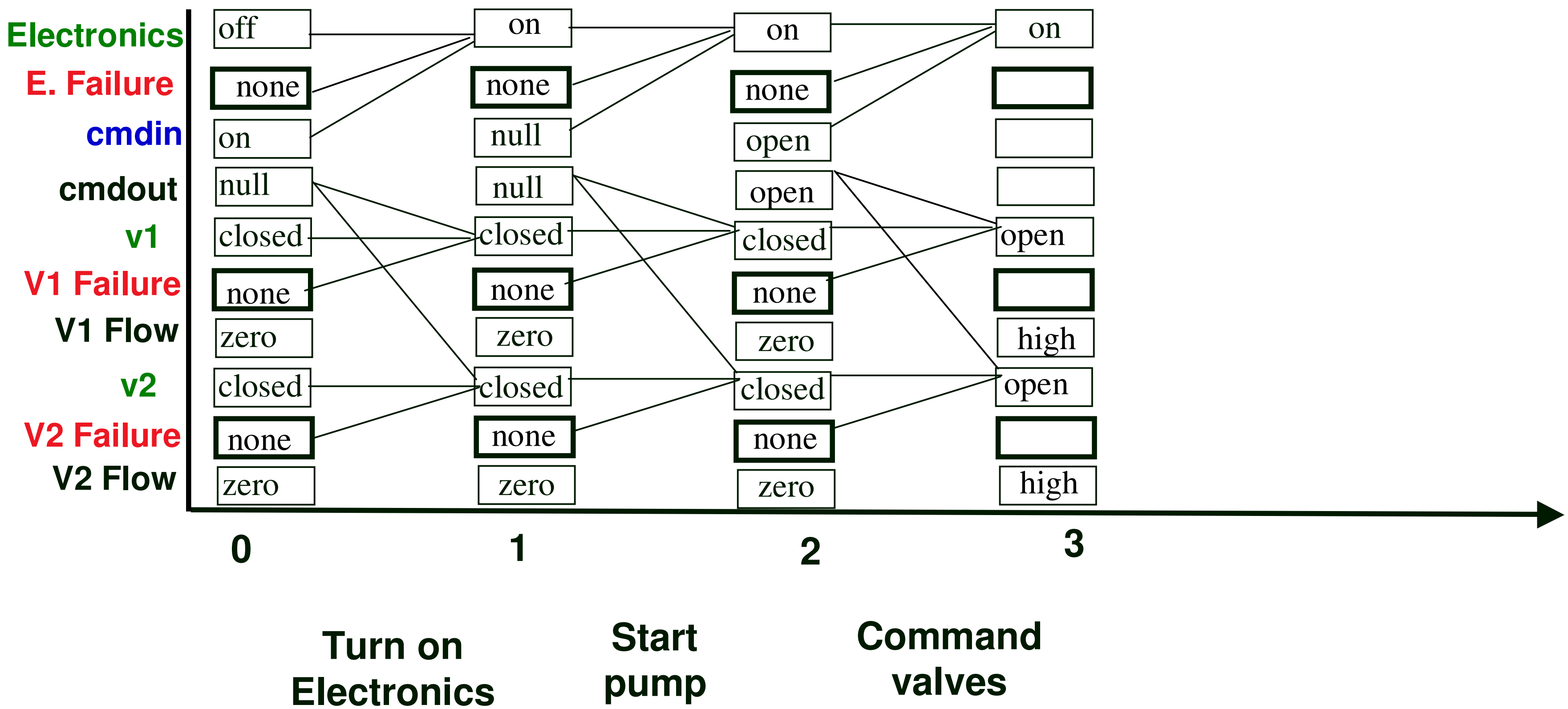
parc
Palo Alto Research Center

Trajectory Representation



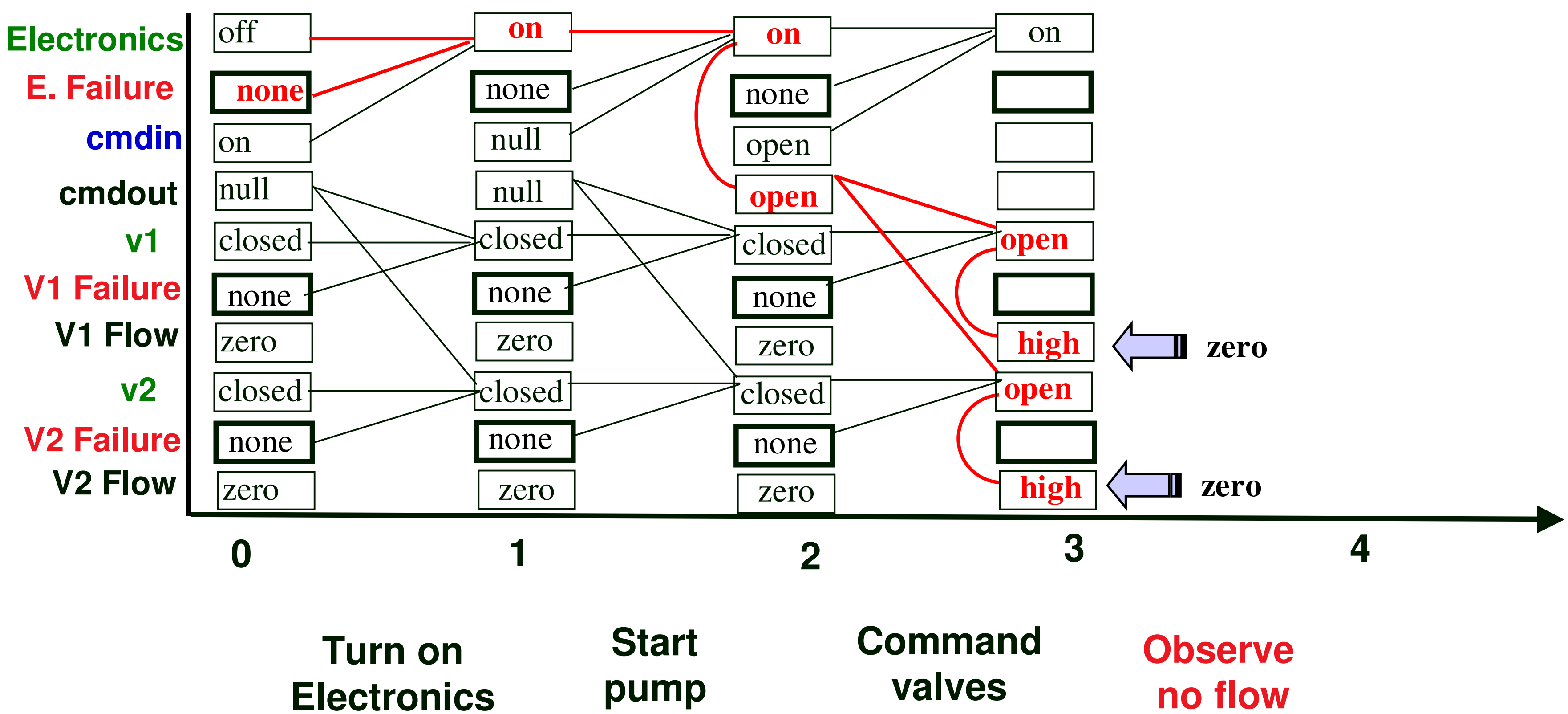
parc Palo Alto Research Center

Trajectory Representation



parc Palo Alto Research Center

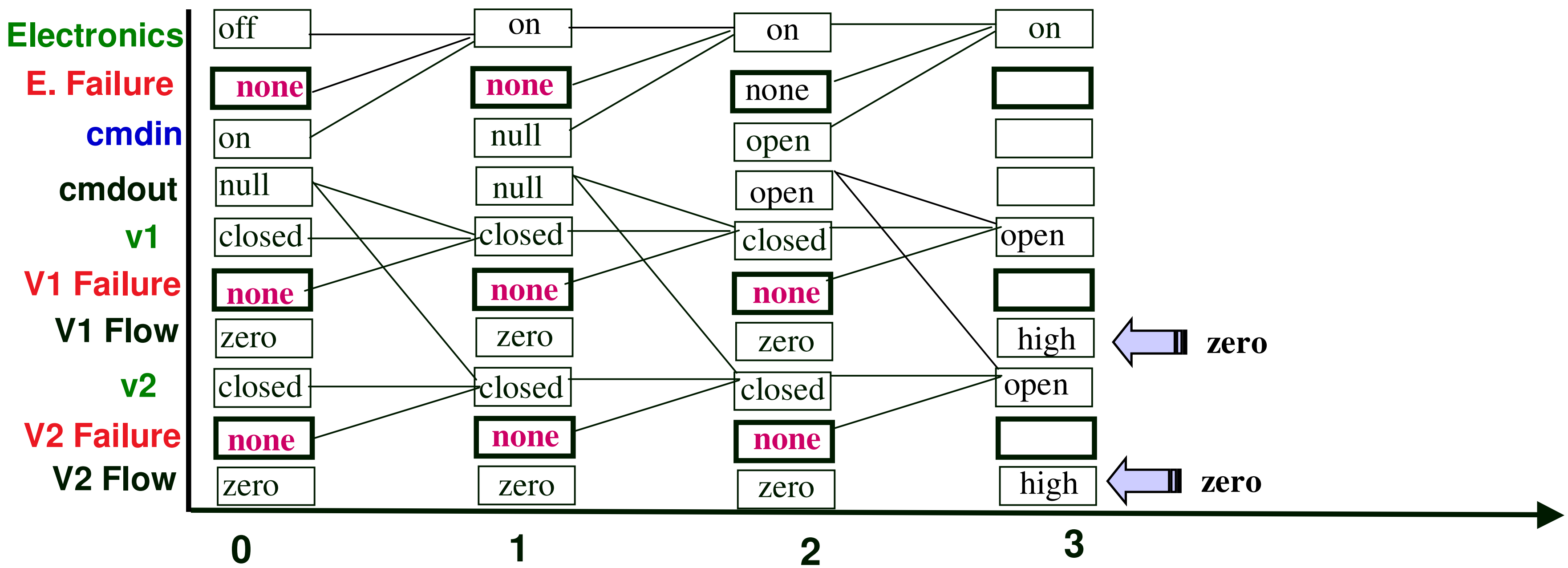
Trajectory Representation



parc Palo Alto Research Center

Generating No Goods

No Good: An assignment that conflicts with observations

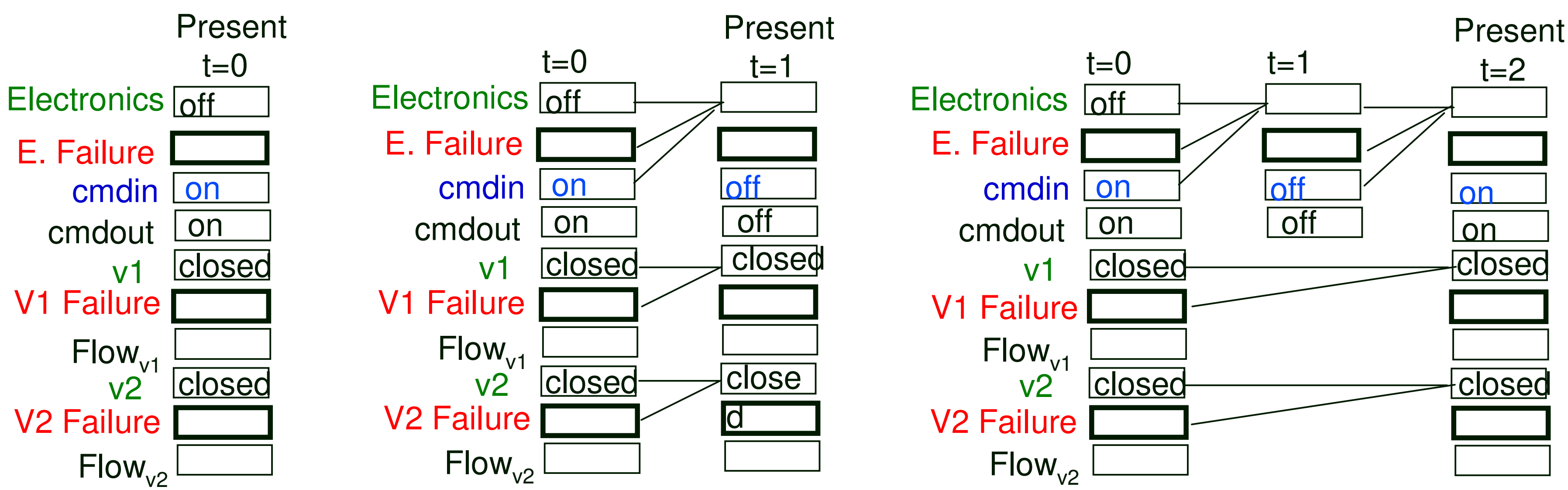


- Every superset of a No Good is implicitly ruled out
- The most likely diagnosis differs from every No Good
- We can use conflict-based search (de Kleer & Williams 1989)

parc Palo Alto Research Center

Minimizing Each Time Step

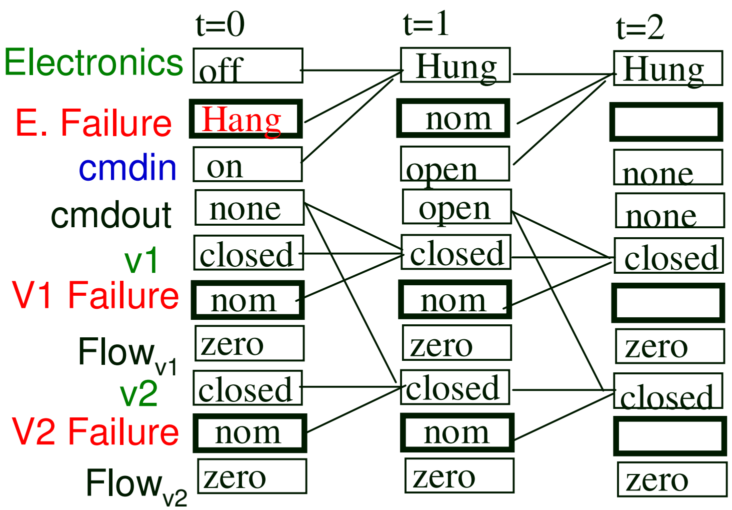
- Intuition: Many temporal distinctions are irrelevant
- Leverage: Merge times *t* and *t-1* for for irrelevant variables

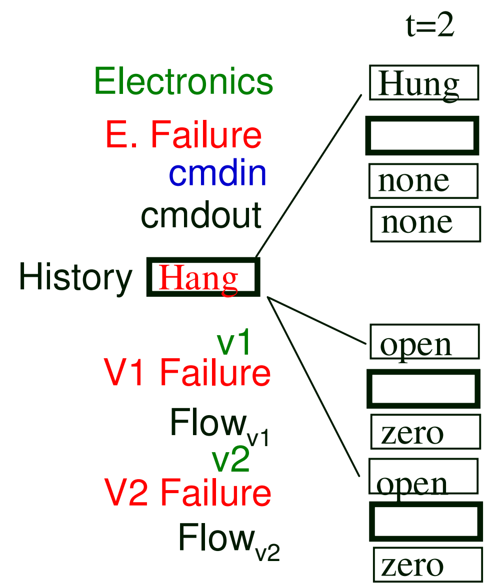


- Proven to be a conservative approximation

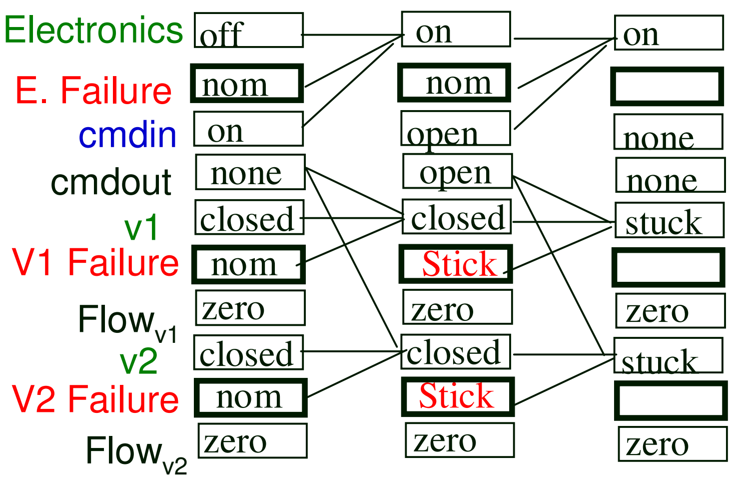
parc
Palo Alto Research Center

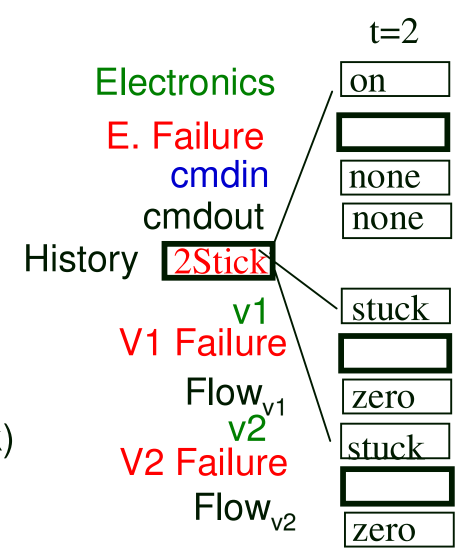
Truncating the Representation





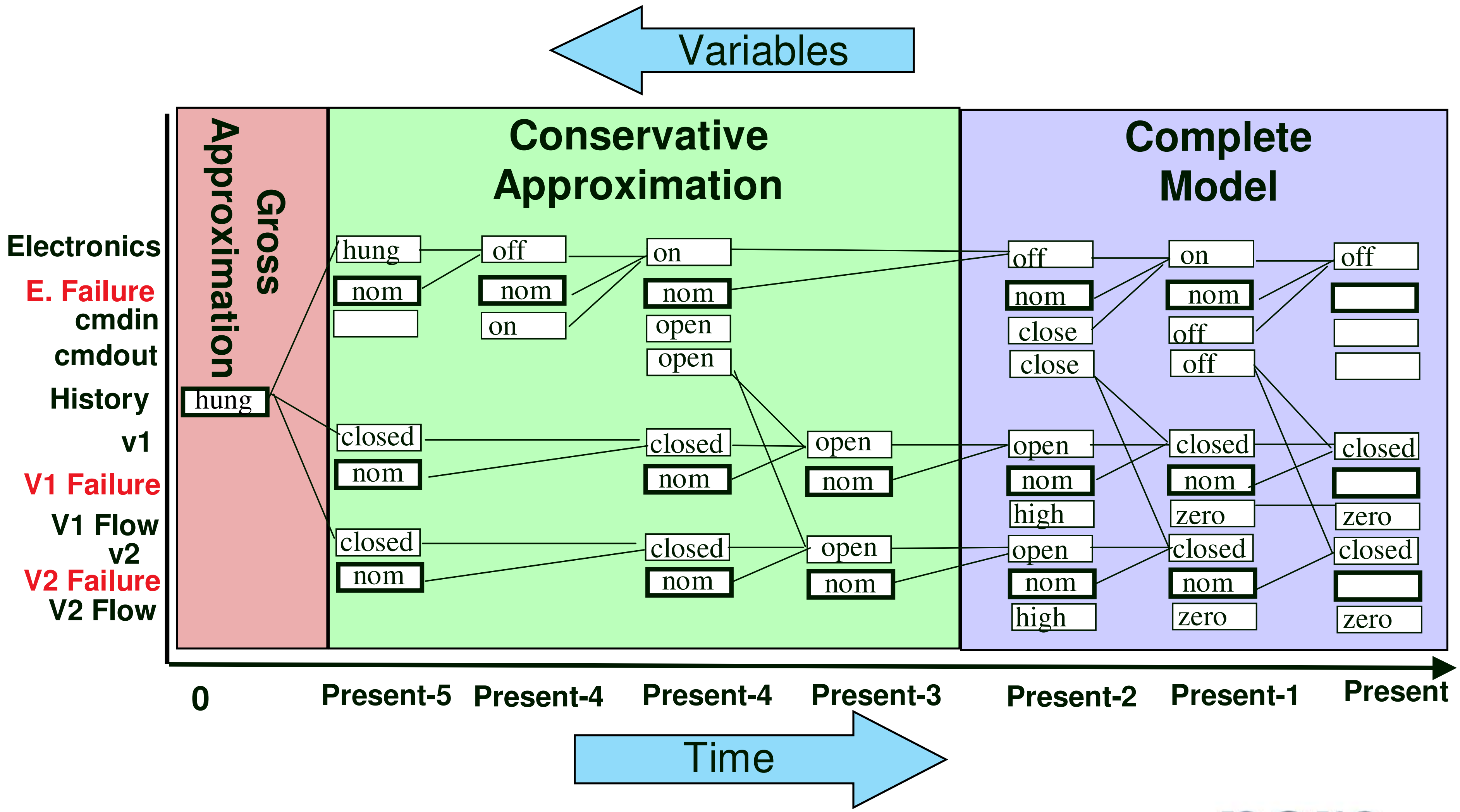
P(History=hang)=
P(hang)





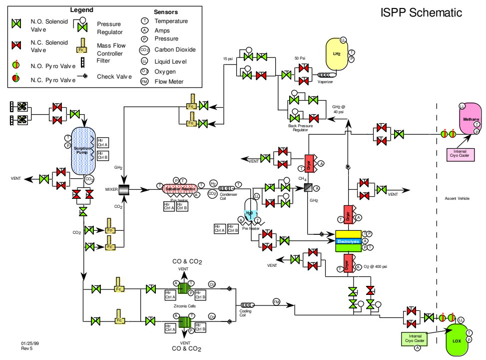
P(History=2stick)=
P(stuck)*P(stuck)

Complete Representation



parc
Palo Alto Research Center

L2 Contributions in Diagnosis

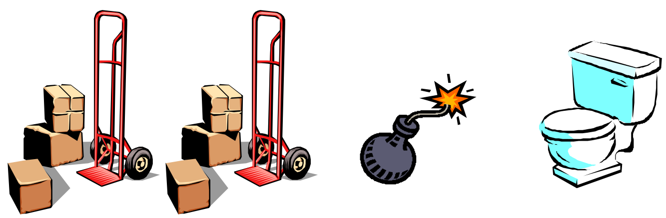


- Developed novel representation for diagnosis over time
- Demonstrated low growth and also constant sized approximations
- Developed novel algorithm for finding all same-probability diagnoses
- Results published in Kurien and Nayak, AAAI 2000
- Significant real-world validation performed
  - Engineers modeled the X34 and X37 and ran diagnostic scenarios
  - Available for non-profit use and for-profit licensing from NASA

parc
Palo Alto Research Center

Performance on Bomb in the Toilet Problems

Packages	Toilets	FragPlan	HSCP	GTP	CMBP
6	1	0.11	0.01	0.07	0.01
8	1	0.47	0.01	0.11	0.20
10	1	2.89	0.01	1.31	0.71



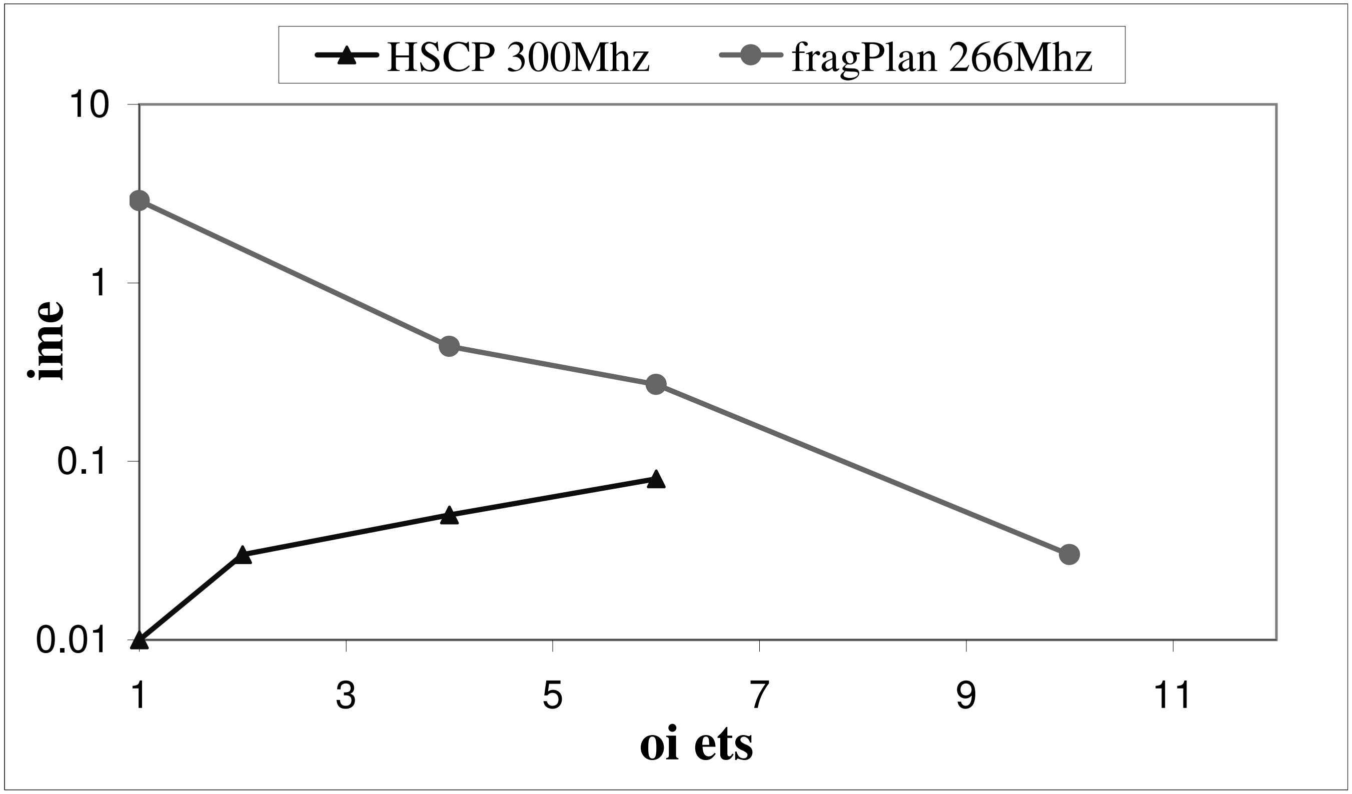
- HSCP dominates on serial (single toilet) instances
- FragPlan is competitive with GTP, CMBP

Bomb in the Toilet with Parallelism

Packages	Toilets	FragPlan	HSCP	GTP	CMBP
8	1	0.47	0.01	0.11	0.20
8	4	0.23	0.04	8.78	2.74
8	6	0.05	0.08	68.43	20.71

- HSCP, CMBP, GPT cannot produce parallel plans
- They produce much longer, harder, serial plans
- Only FragPlan & C-Plan (not shown) are truly parallel
- C-Plan fails on most serial instances

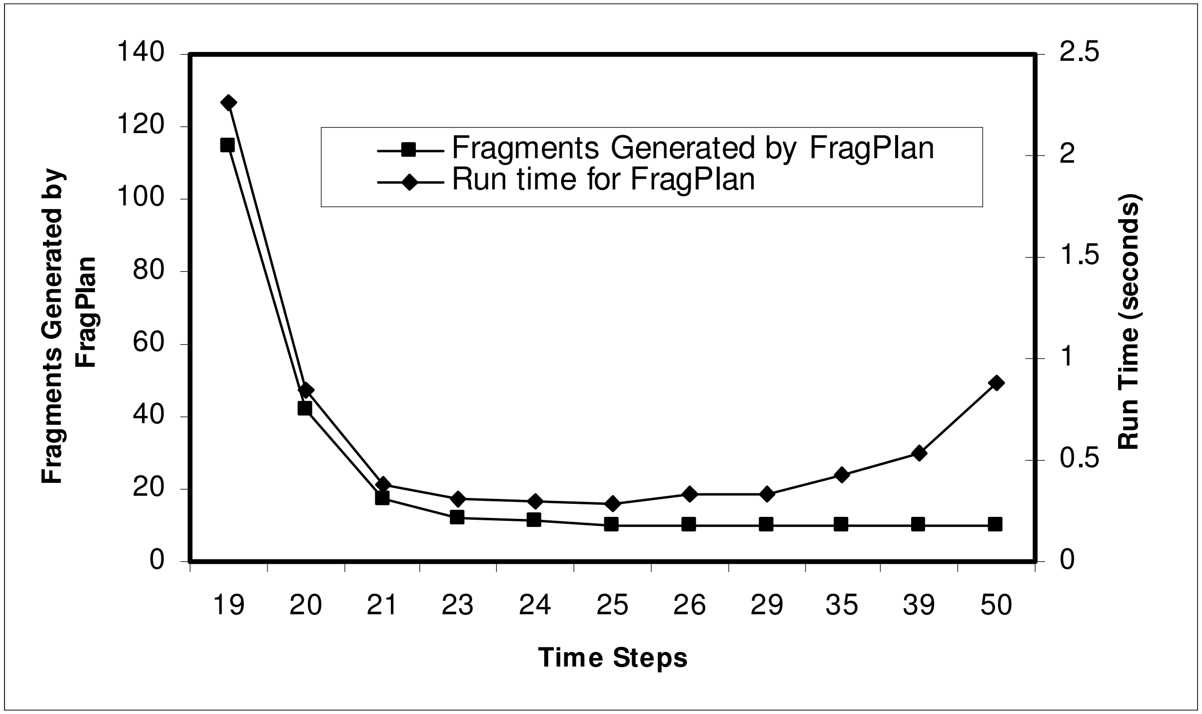
Bomb in the Toilet with Parallelism



- Space of serialized plans explodes as parallelism increases
- Fragments become independent, yielding linear speedup

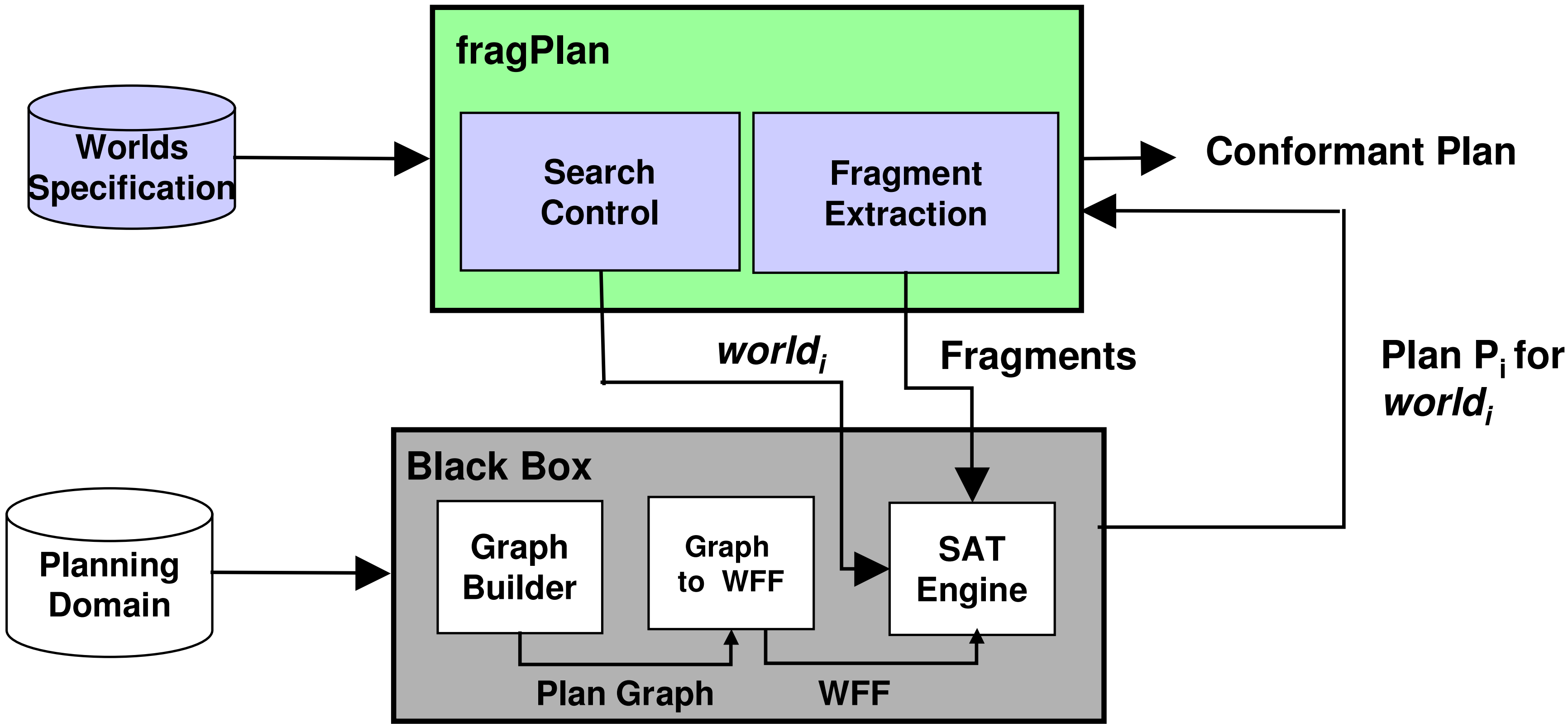
parc
Palo Alto Research Center

Planning with Extra Time Steps



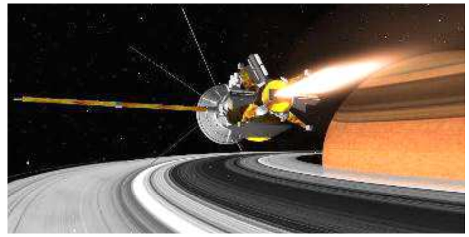
- For fragPlan, density of conformant plans rises
- For other planners, search depth grows

Implementation



- We currently use Black Box (Kautz & Selman 99) as a black box
- Black Box encodes the problem as propositional satisfiability
- Randomized SATZ used to find an assignment (*i.e*. a plan)

Is Conformant Planning Enough?



- No conformant plan may exist due to failures
    - Some goals may not be achievable in any world
    - Some possible worlds may not allow all goals
- Certain actions may violate safety constraints
    - Safety always desired, often dominates
    - Certain goals dominate at critical junctures
    - A failure may force all actions to be unsafe
- Time for planning not known a priori
    - We must have some plan

Given: Partial ordering on goals, safety, and worlds
Return: Best plan the available time allows

parc Palo Alto Research Center

SCOPE – Safe, Conformant, Optimizing Planning Engine

- Approach: Manipulate the scope of the problem

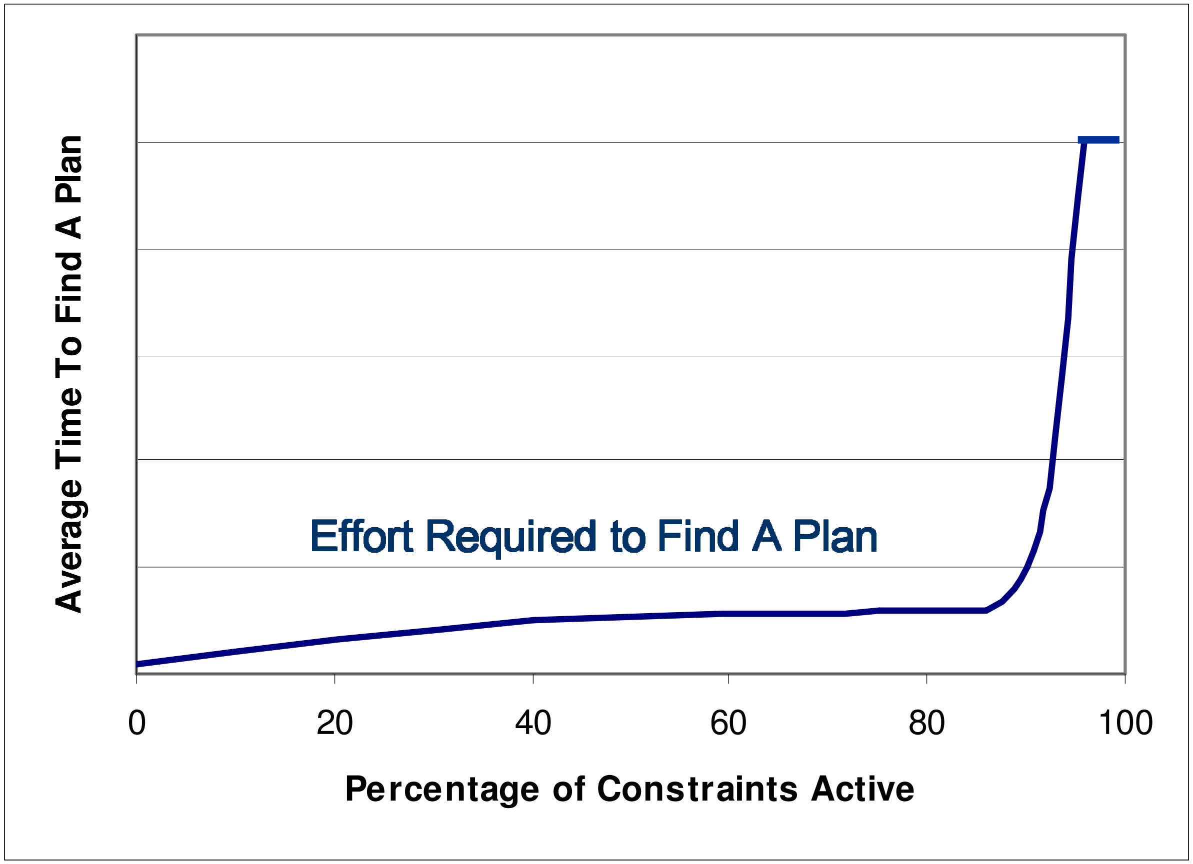
While ($Time \neq 0$) {
  select constraints from $\{Worlds \cup G \cup S\}$
  FragPlan(constraints) for some time
  }

- Pareto-optimality requires checking all constraint subsets

- We have developed many simpler selection policies and experimented with several

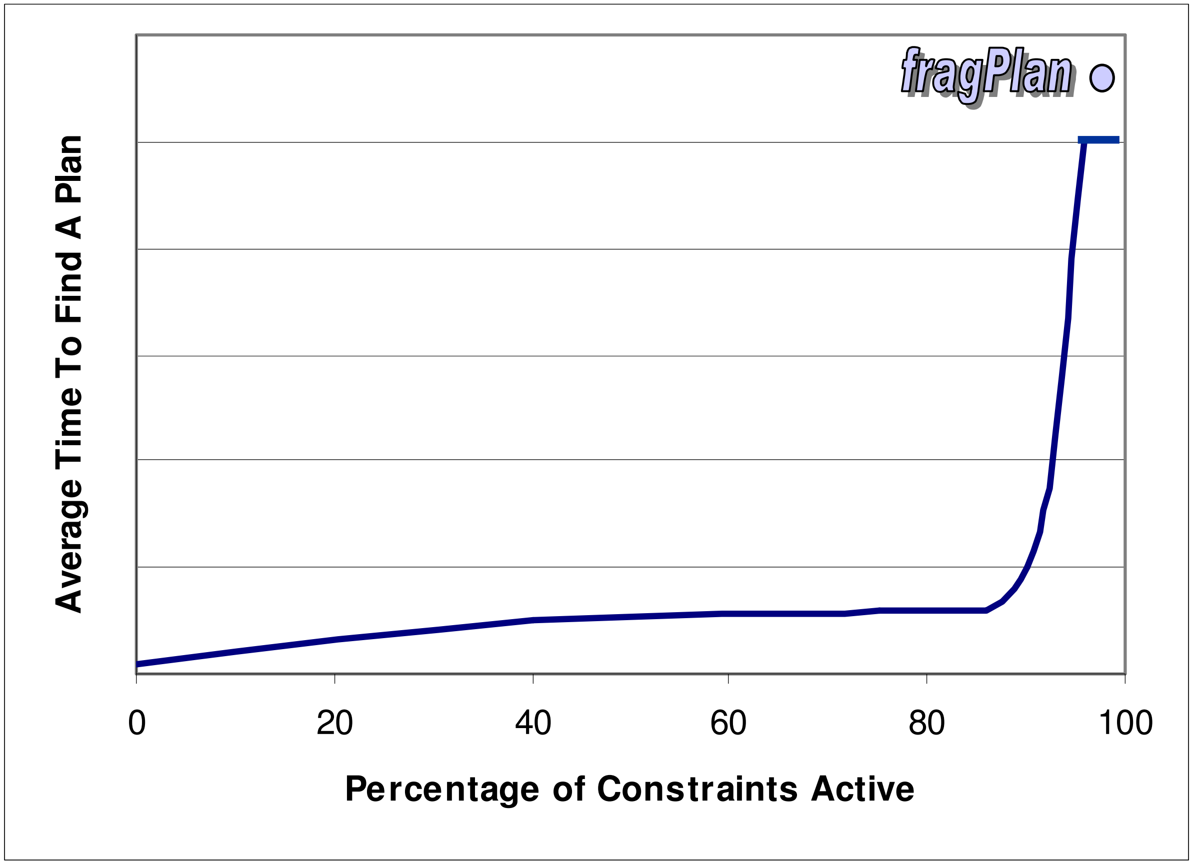
parc Palo Alto Research Center

Typical Planning Problem Difficulty



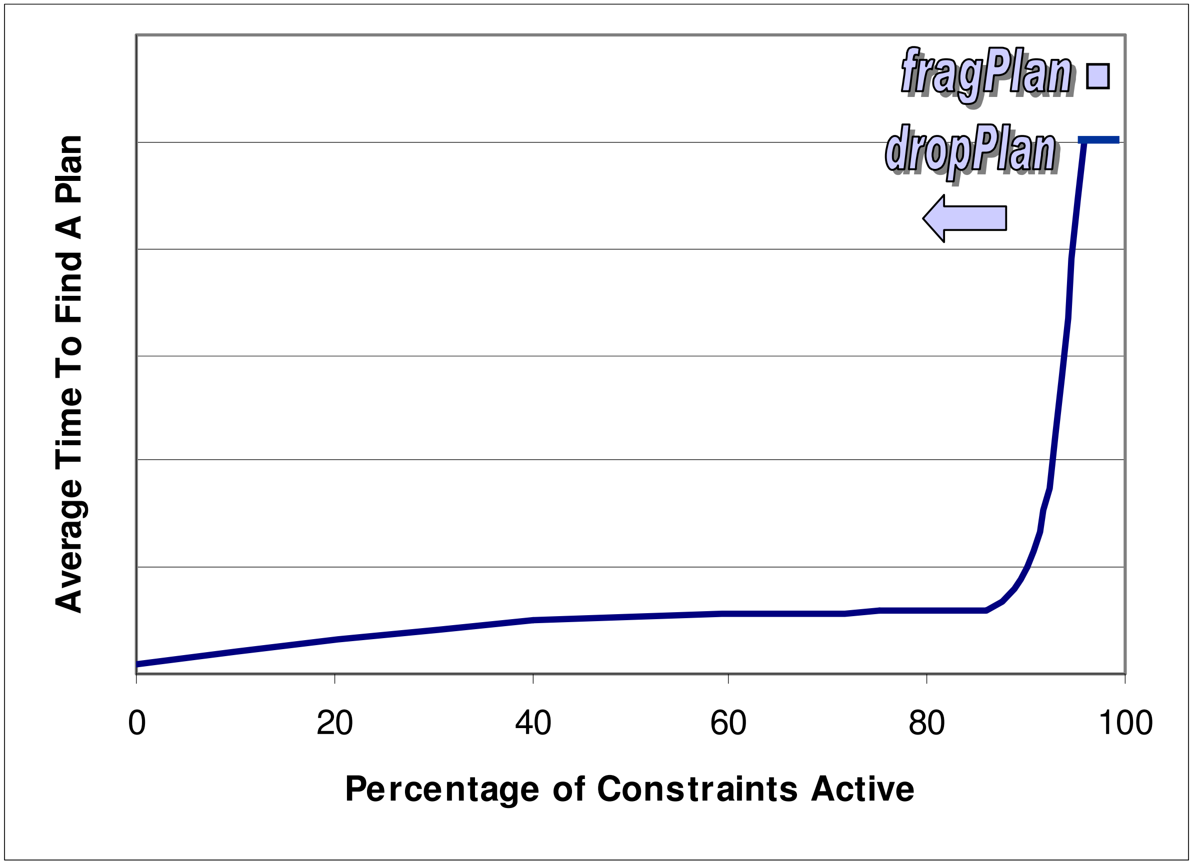
parc Palo Alto Research Center

SCOPE Strategies



- fragPlan Strategy: Go for broke
  - Devote all time to solve entire constraint set

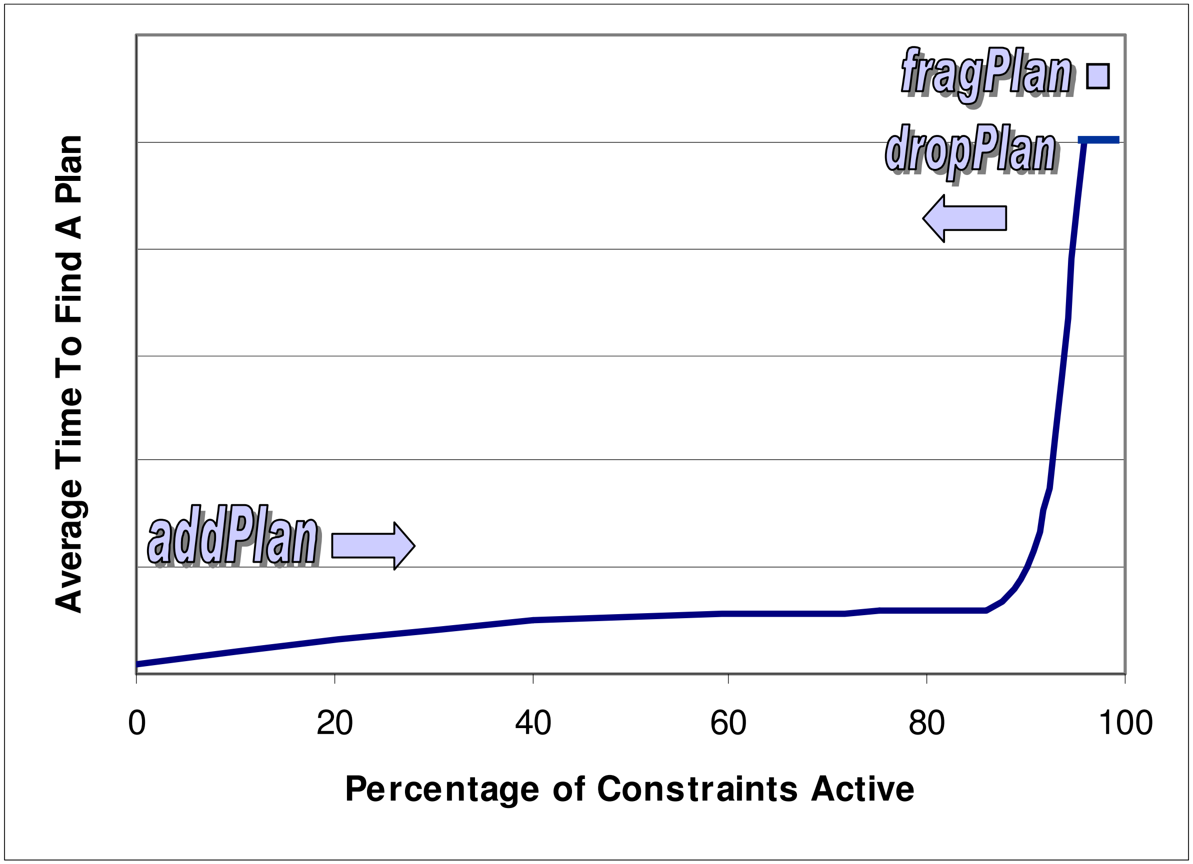
SCOPE Strategies



- dropPlan Strategy: Start big, shrink
  - Devote 1/n of remaining time to solve entire constraint set
  - Failure reveals difficult constraint combinations
  - On failure, remove constraints guided by O, difficulty

parc Palo Alto Research Center

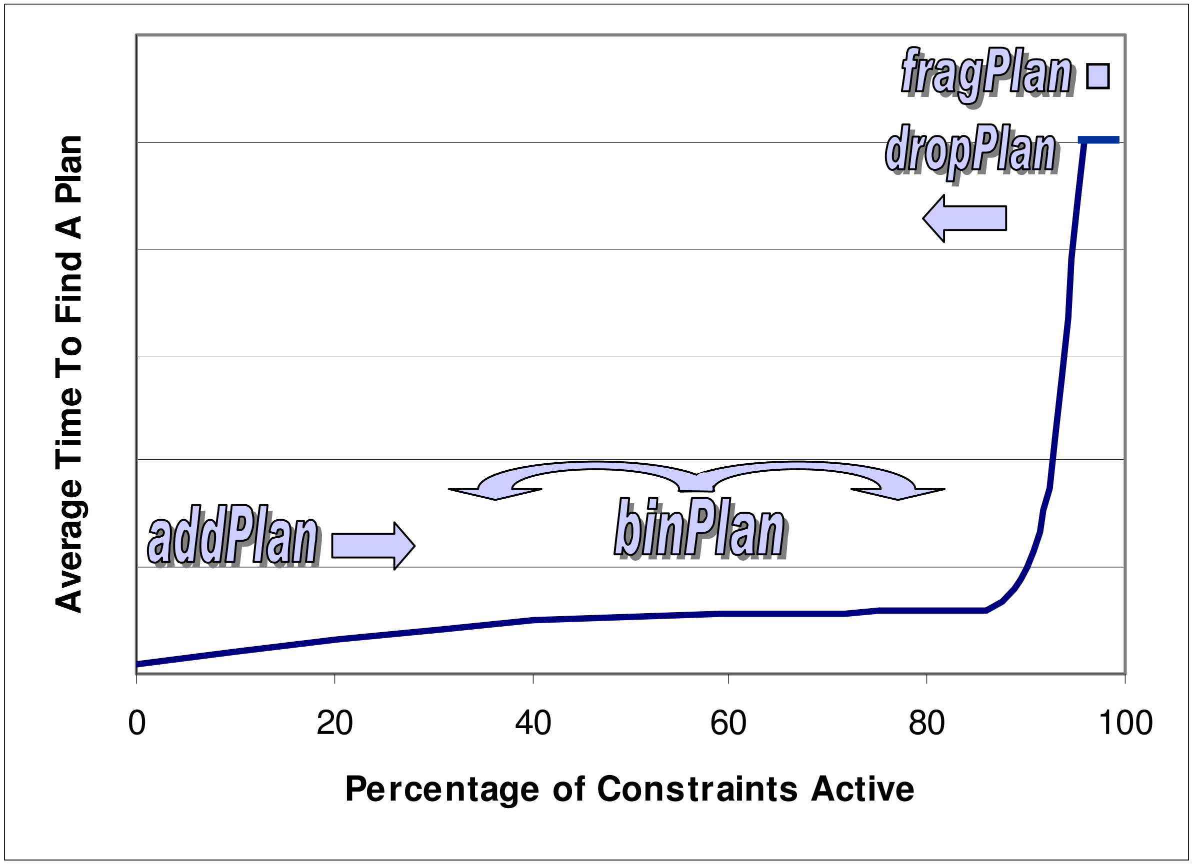
SCOPE Strategies



- addPlan Strategy: Start small and grow
  - Devote all remaining time to solving simplest problem
  - Anytime
  - On success, add constraints guided by partial ordering O
  - Successful plans used to seed next planning attempt

parc
Palo Alto Research Center

SCOPE Strategies

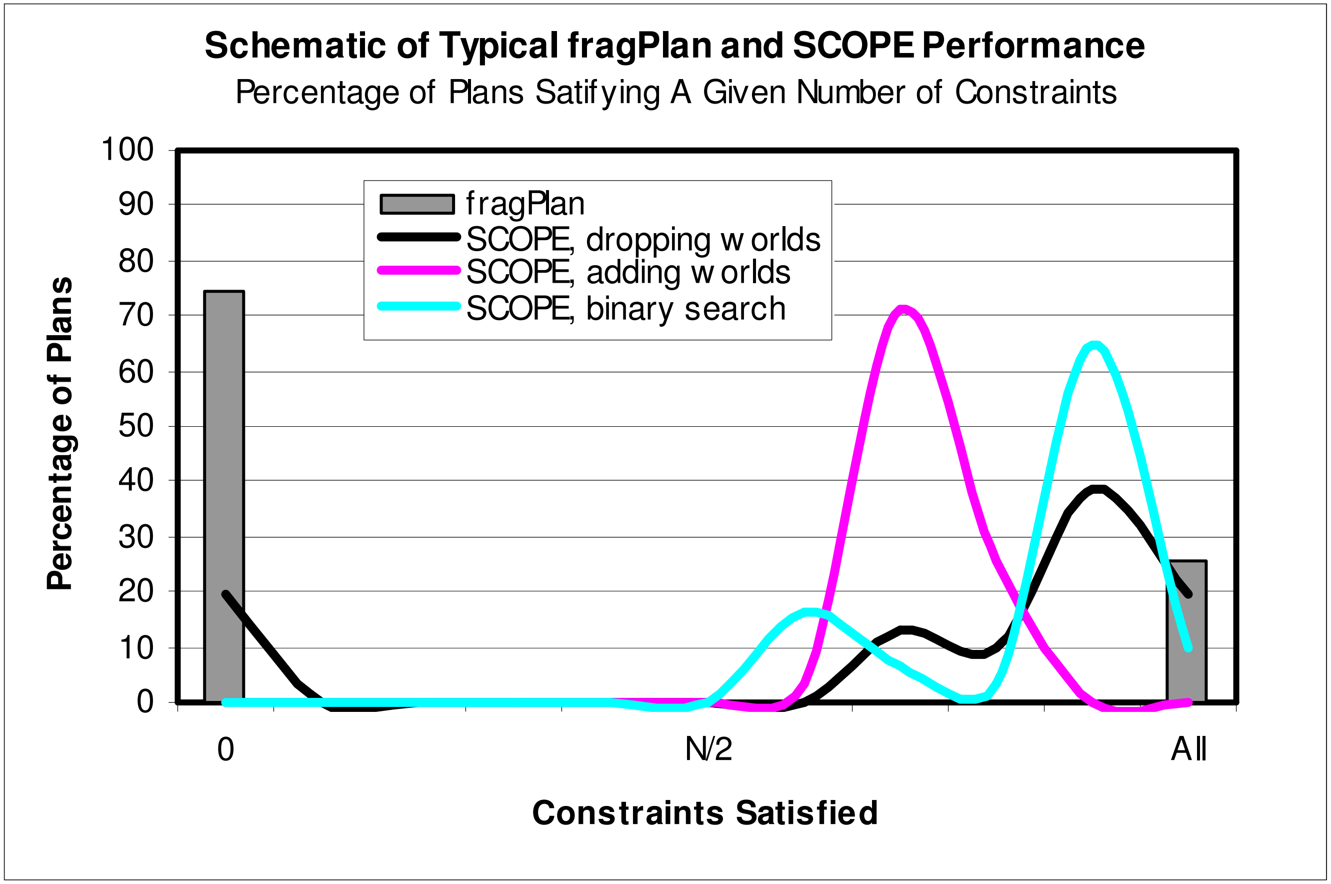


- binPlan Strategy: Start in the middle, grow or shrink
  - Attempts to rise faster than addPlan, fail less than dropPlan

parc
Palo Alto Research Center

Some Observations On SCOPE

- fragPlan produces more conformant plans
- All SCOPE variations have better expected performance

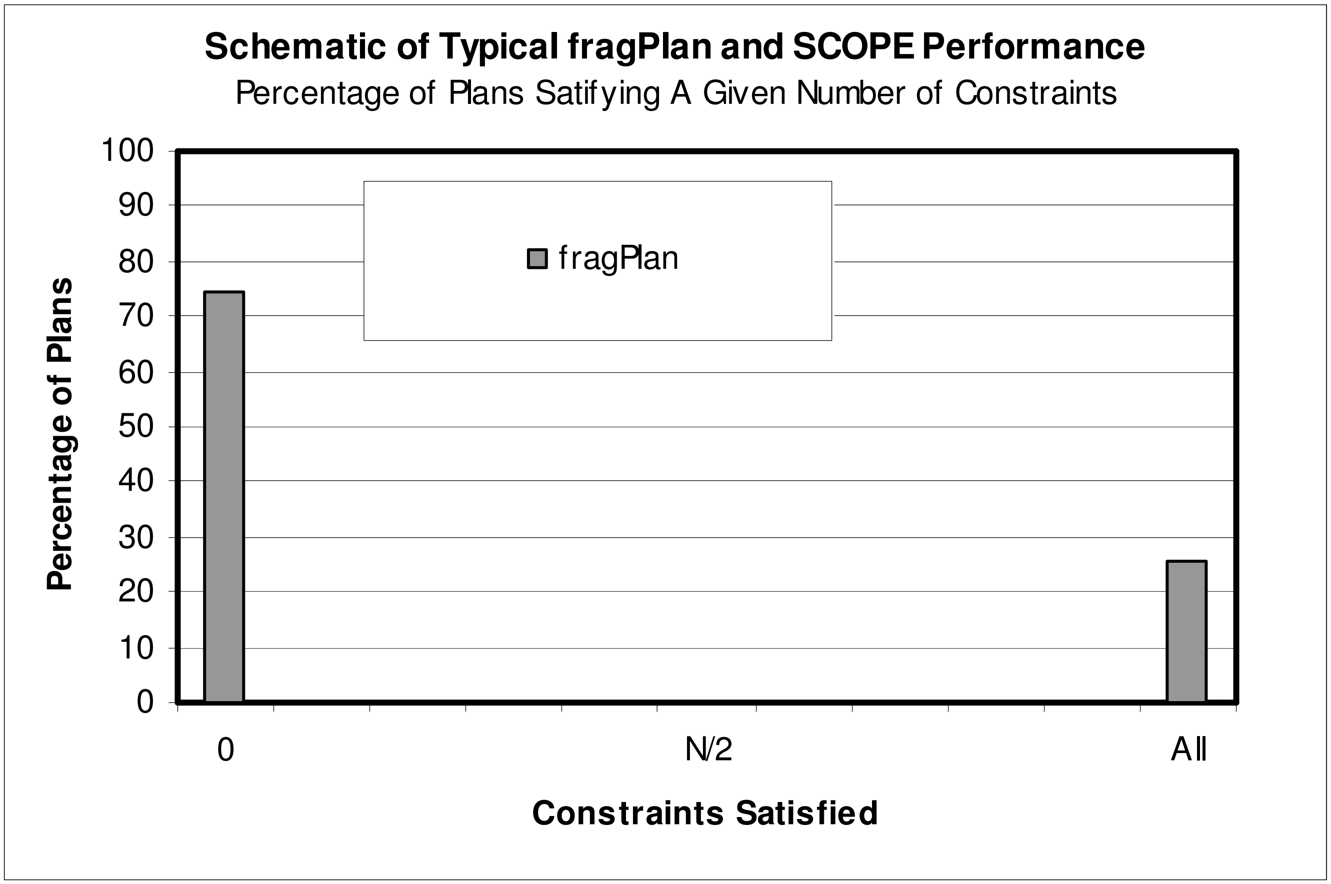


Other Strategies

- We want the problem that is just short of too hard
- Intuitively, we attempt to learn the difficulty curve
- dropPlan with difficulty
  – When a plan fails, we can often identify the constraints at fault
  – We remove constraints that consistently cause problems
- addPlan with sliding
  – If the first plans are easy, move right faster
- Reversal of fortune
  – Start with all constraints and drop, learning difficulty
  – If time runs short, drop all constraints and add least difficult

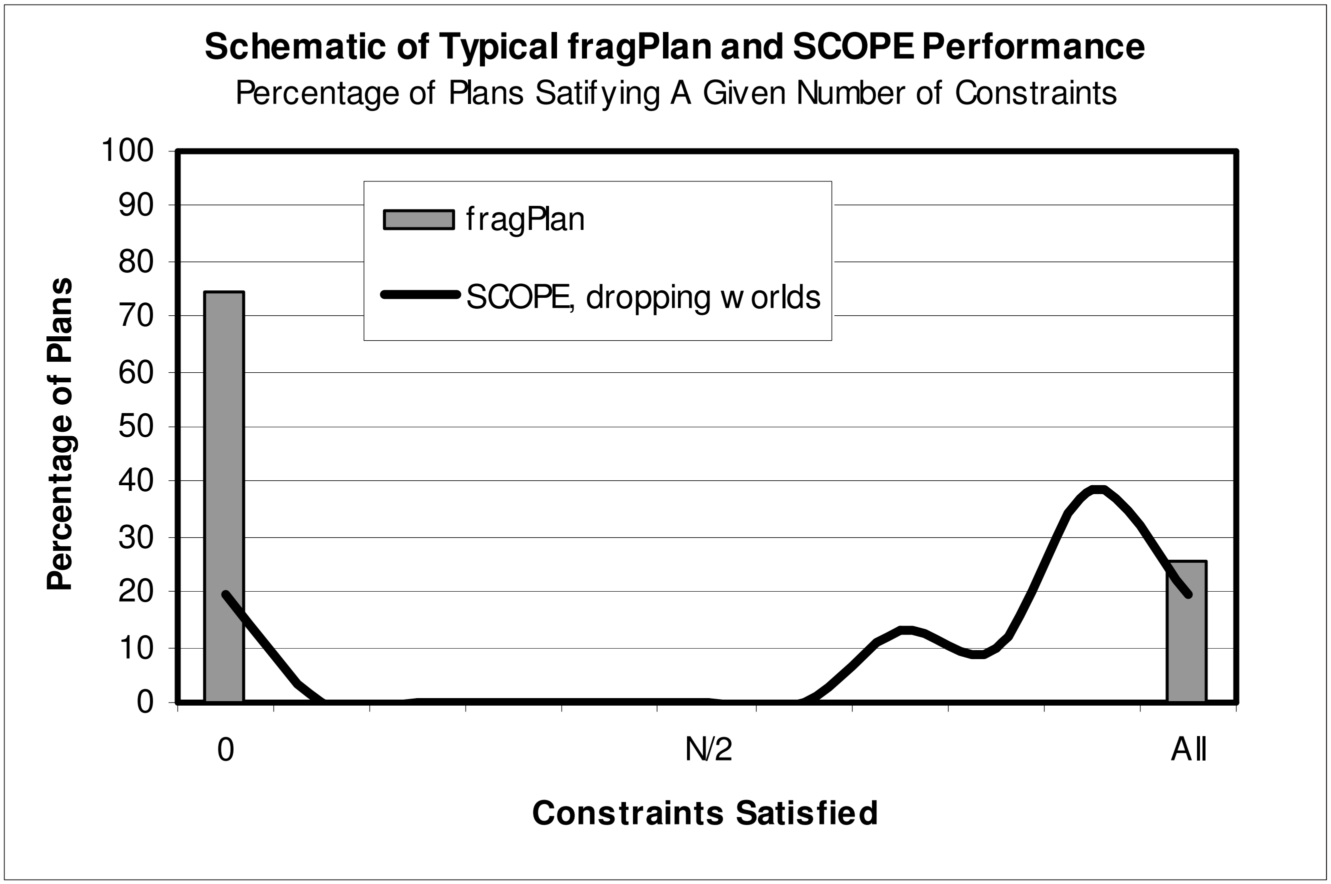
parc Palo Alto Research Center

Typical Performance of fragPlan



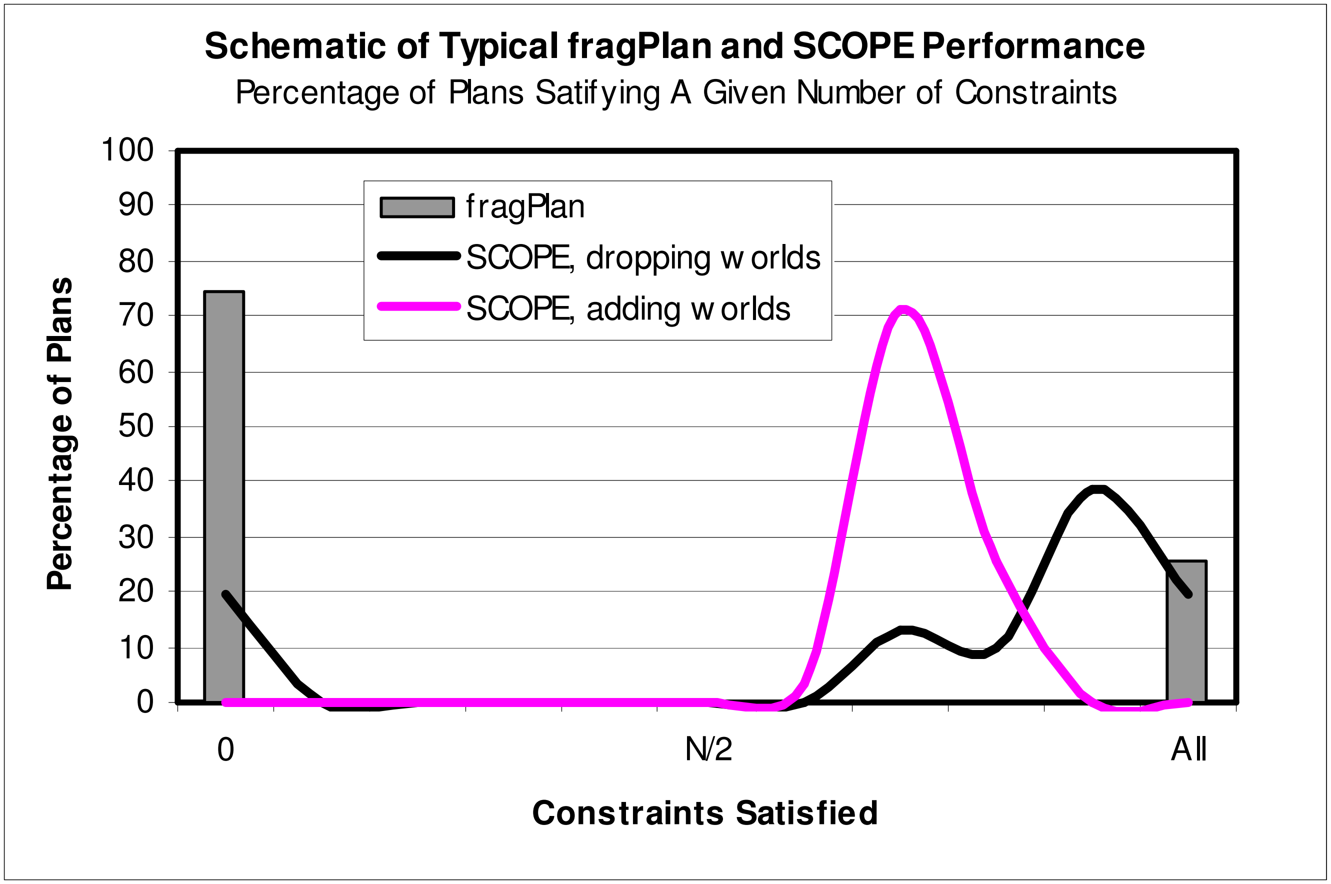
parc
Palo Alto Research Center

Typical Performance of fragPlan vs SCOPE



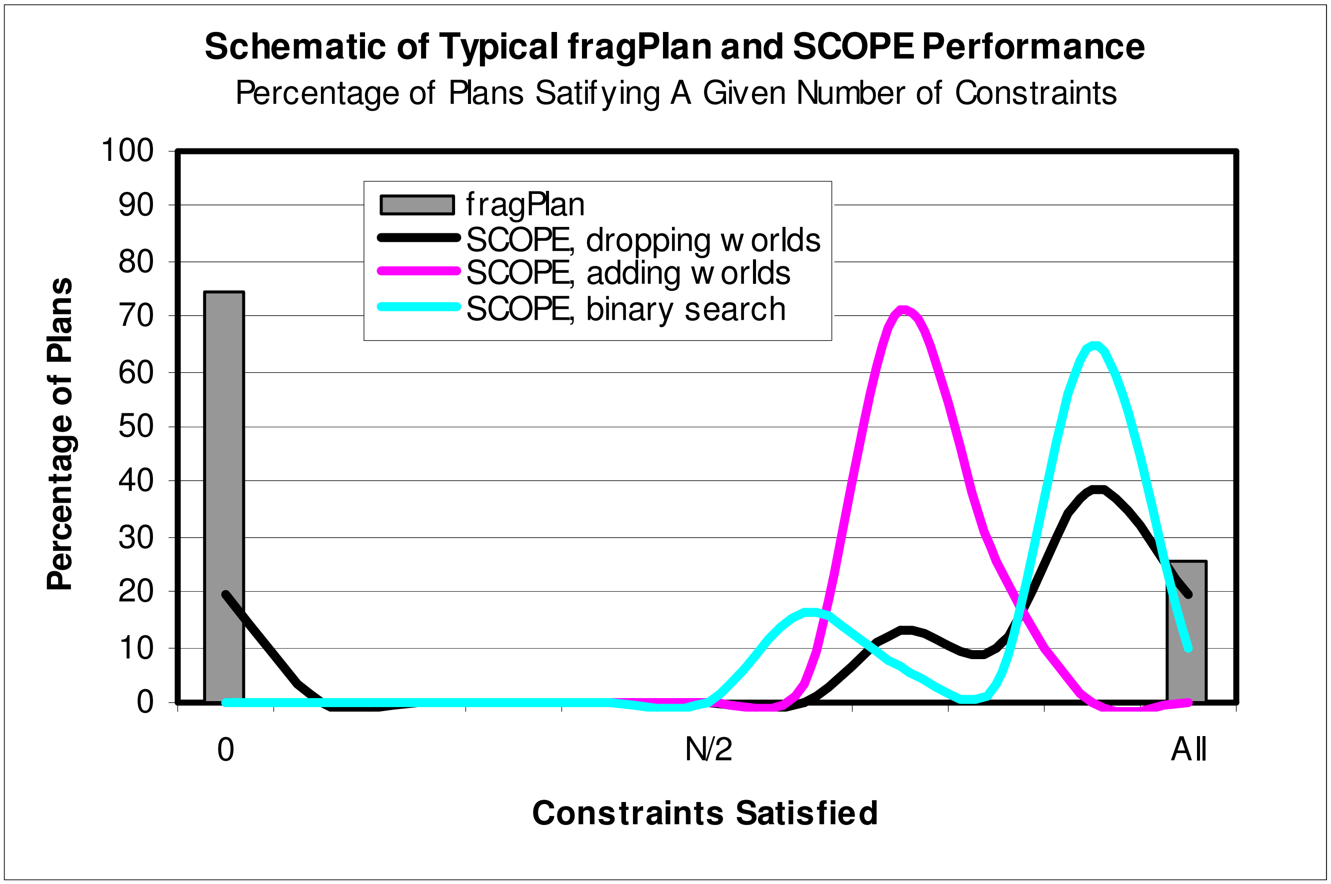
parc
Palo Alto Research Center

Typical Performance of fragPlan vs SCOPE



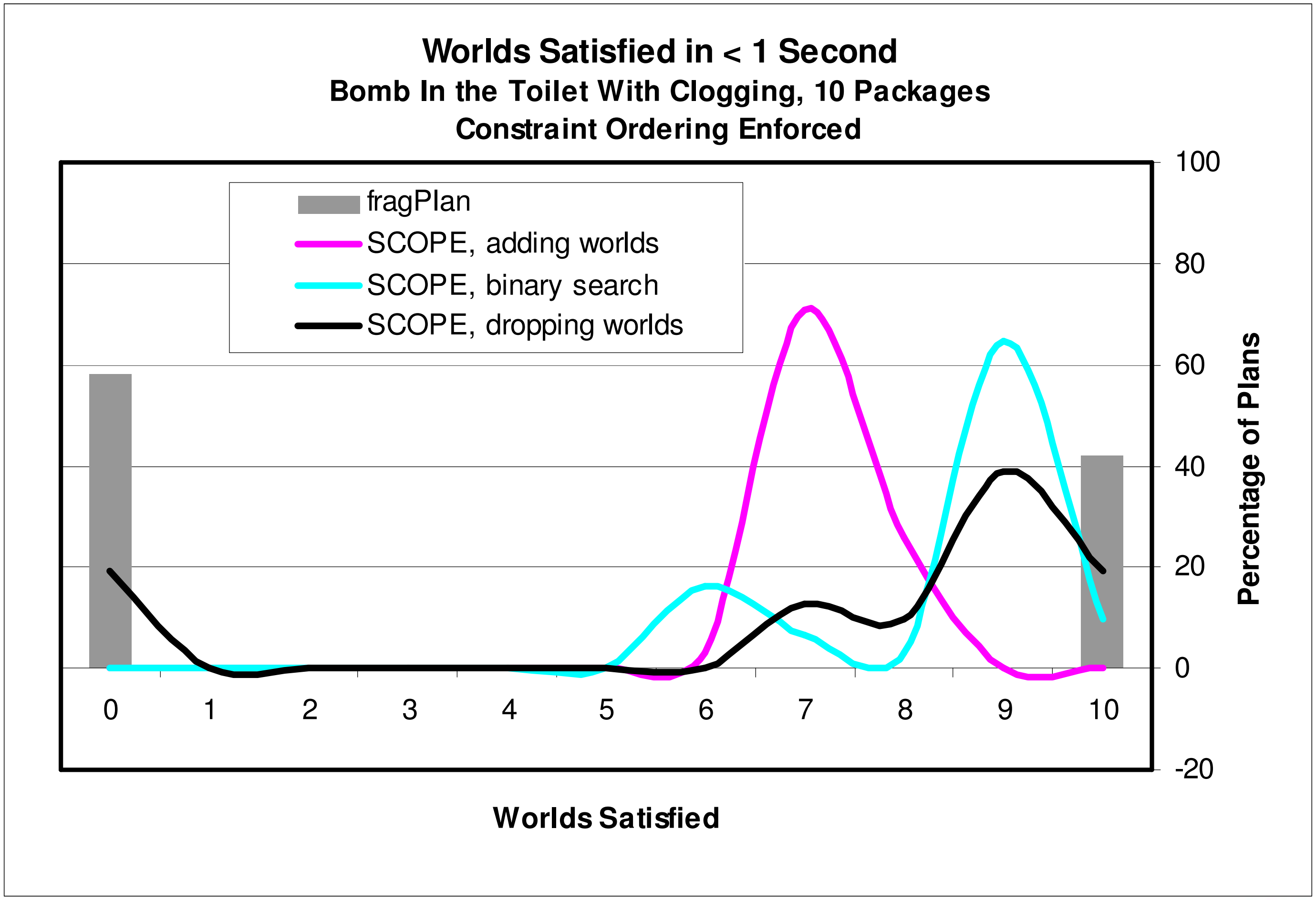
parc Palo Alto Research Center

Typical Performance of fragPlan vs SCOPE



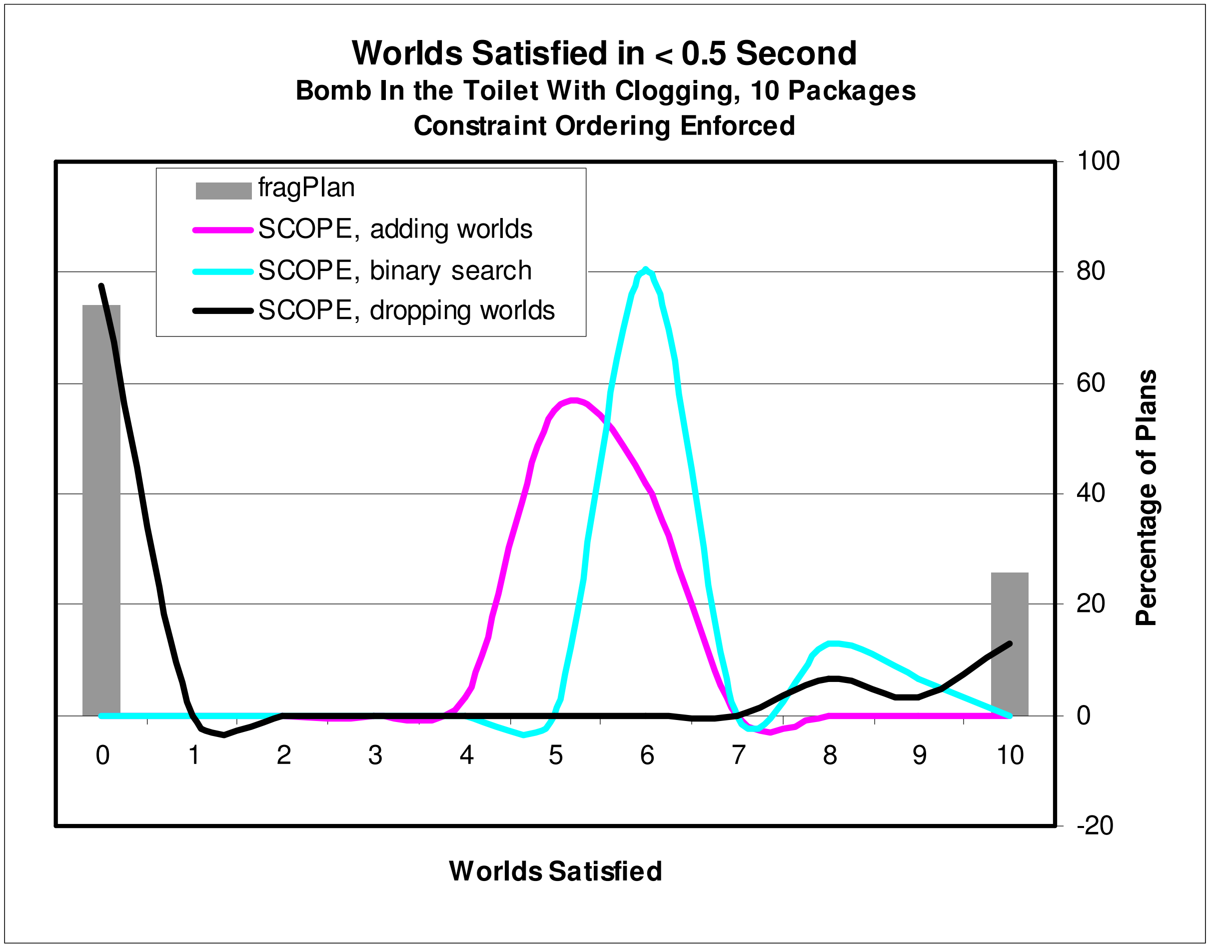
parc
Palo Alto Research Center

Strategy Performance Versus Time



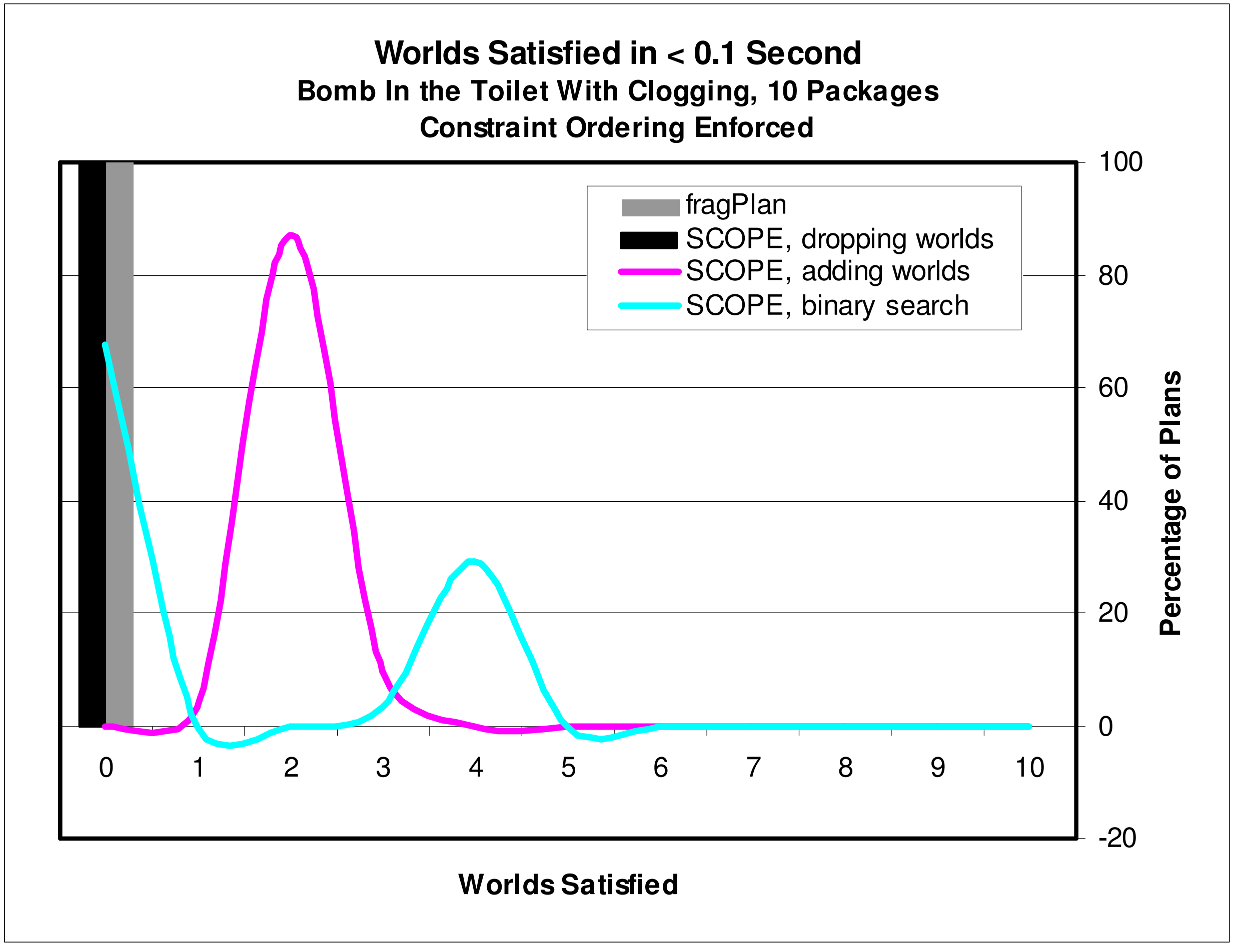
parc
Palo Alto Research Center

Strategy Performance Versus Time



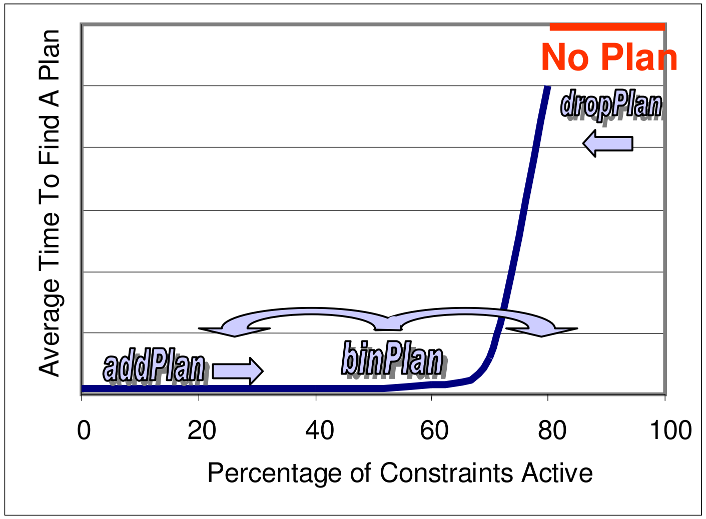
parc Palo Alto Research Center

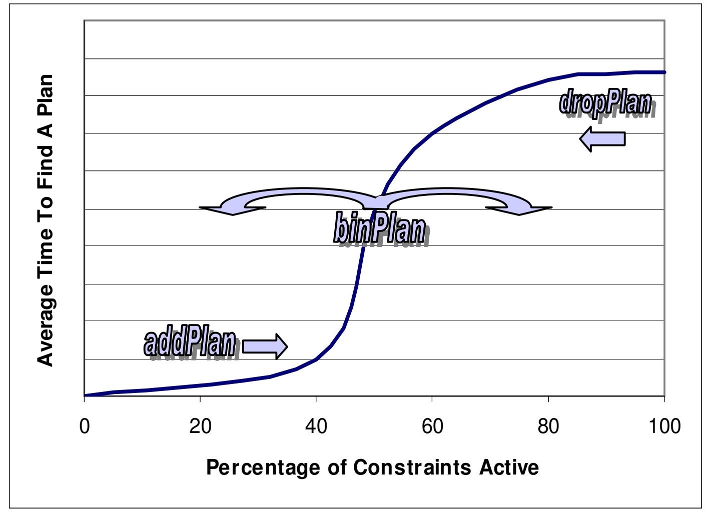
Strategy Performance Versus Time



parc Palo Alto Research Center

Other Strategies



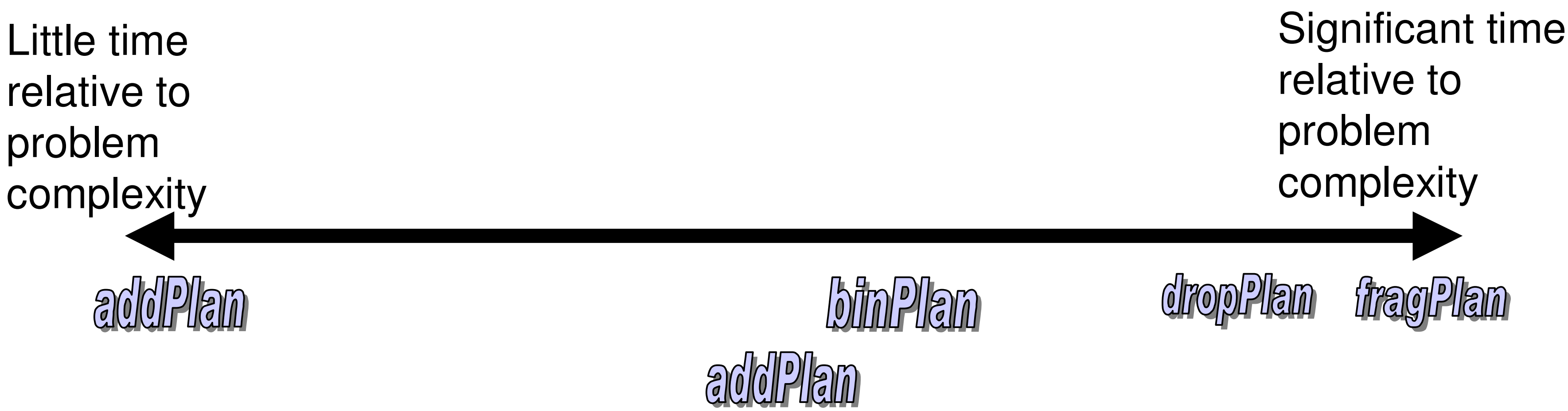


- dropPlan with difficulty
  - When a plan fails, we can identify the constraints at fault
  - We remove constraints that consistently cause problems
- addPlan with sliding
  - If the first plans are easy, move right faster
- Reversal of fortune
  - Start with all constraints and drop, learning difficulty
  - If time runs short, drop all constraints and add least difficult

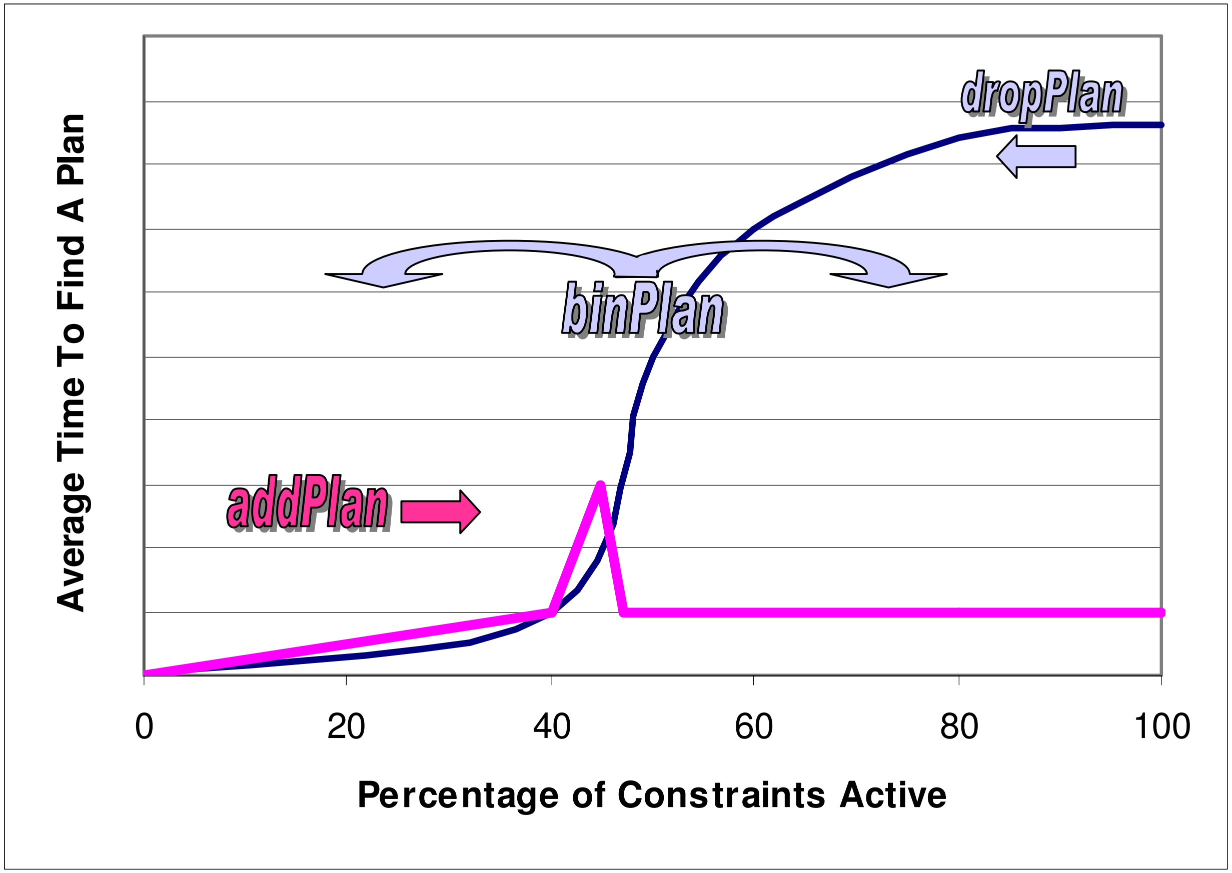
parc Palo Alto Research Center

Some Observations On SCOPE

- The best SCOPE strategy varies with time, problem



Some Observations On SCOPE



- addPlan has the advantage of extending an existing plan